TECHNICKÁ UNIVERZITA V LIBERCI

Fakulta mechatroniky a mezioborových inženýrských studií

ústav nových technologií

a aplikované informatiky

Diplomová práce

Pokladní systém pro restaurace a bary

POS system for restaurants and bars

Bc. Martin Hozák

Vedoucí diplomové práce: Mgr. Jiří Vraný, Ph.D.

Konzultant Stanislav Šilhavý

Studijní program: N2612 Elektrotechnika a informatika

Studijní obor: 1802T007 - Informační technologie

Květen 2013

Tady bude oficiální zadání.

Poděkování

Na tomto místě bych chtěl poděkovat Mgr. Jiřímu Vranému, Ph.D. za vedení diplomové práce a za užitečné rady při psaní tohoto dokumentu. Mé poděkování patří také Stanislavu Šilhavému za rady a připomínky při tvorbě praktické části.

Největší poděkování však patří mým rodičům, za podporu během mého studia.

Prohlášení

Byl(a) jsem seznámen(a) s tím, že na mou diplomovou práci se plně vztahuje zákon č. 121/2000 Sb. o právu autorském, zejména § 60 – školní dílo.

Beru na vědomí, že Technická univerzita v Liberci (TUL) nezasahuje do mých autorských práv užitím mé diplomové práce pro vnitřní potřebu TUL.

Užiji-li diplomovou práci nebo poskytnu-li licenci k jejímu využití, jsem si vědom povinnosti informovat o této skutečnosti TUL; v tomto případě má TUL právo ode mne požadovat úhradu nákladů, které vynaložila na vytvoření díla, až do jejich skutečné výše.

Diplomovou práci jsem vypracoval(a) samostatně s použitím uvedené literatury a na základě konzultací s vedoucím diplomové práce a konzultantem.

V Liberci dne …… Podpis ……..

ABSTRAKT

Tato práce se v první části zabývá teoretickým popisem architektur REST a MVC, použitých návrhových vzorů a základním popisem platformy Android. Součástí práce je rešerše existujících pokladních systémů a porovnání s navrhovaným řešením. Hlavní náplní je samotný návrh a realizace vlastního pokladního systému s detailním popisem zvoleného řešení. Realizace je rozdělena do čtyř logických částí, kde každá z nich odpovídá jedné z aplikací vytvořeného systému.

ABSTRACT

The first part of this presentation deals with the theoretical description of REST and MVC architectures, design patterns and basic description of the Android platform. Component of this work is research of existing POS systems and comparing with the proposed solution. Design and the realization of the POS system with a detailed description of the solution is main contents of this work. Implementation is divided into four logical parts, each of one corresponds to one of the applications created by the system.

Klíčová slova

REST, MVC, JAVA, Android, POS

Keywords

REST, MVC, JAVA, Android, POS

OBSAH

Seznam zkratek

AJAX - Asynchronous JavaScript and XML

API - Application Programming Interface

CSS - Cascading Style Sheets

ER - Entity-relationship

HTML - Hyper Text Markup Language

HTTP - Hypertext Transfer Protocol

JASON - JavaScript Object Notation

MySQL - My Structured Query Language

PHP - Hypertext Preprocessor

SQL - Structured Query Language

REST - Representational State Transfe

Seznam obrázků

1. Úvod

V dnešní době se můžeme s výpočetní technikou setkat stále častěji i na místech, kde by to dříve bylo nemyslitelné převážně kvůli vynaloženým nákladům.

Jedním z příkladů je i využití výpočetní techniky v pohostinství. S její pomocí lze zvýšit kvalitu a rychlost nabízených služeb zákazníkovi a zároveň ušetřit náklady majiteli. Společnost New Prague Media s.r.o. by ráda rozšířila svou nabídku služeb v tomto odvětví a nabízela tak kompletní software pro chod a správu podniku. Z tohoto důvodu jsem byl osloven, zda bych nezpracoval toto téma v rámci své diplomové práce.

Cílem této práce tedy je seznámit se s řešením, které již na trhu existuje. Pomocí těchto znalostí navrhnout a následně vyvinou základní prototyp systému, který bude sloužit pro testování technologií a myšlenek. Systém by z tohoto důvodu měl být co nejuniverzálnější, pokud možno bez vazby na určitou platformu a umožnit tak využít různá zařízení bez nutnosti vynakládání prostředků za specializovaný hardware.

2. Cíle diplomové práce

Stěžejním cílem diplomové práce je vytvořit komplexní balík aplikací, který by umožnil přejít hypotetickému podniku z ručně psaných objednávek na kouscích papíru na elektronickou formu. Hlavní aplikace bude sloužit provozovateli podniku, ve které bude spravovat nabízený sortiment. Zároveň mu bude umožňovat udržovat si přehled o tržbách a aktuální obsazenosti podniku. Tato aplikace bude vytvořena jako webové stránky, které budou dostupné z internetu. Součástí bude API rozhraní, prostřednictvím kterého budou komunikovat aplikace využívané personálem v podniku. V první řadě se bude jednat o pokladnu, ke které bude připojena tiskárna na účtenky a lístky s objednávkou jídel určených do kuchyně. Dále to bude mobilní číšník. Tato aplikace umožní personálu prostřednictvím tabletu, který budou nosit u sebe, vyvářet objednávky pro jednotlivé zákazníky bez nutnosti odbíhání a předávání informací např. o objednaných jídlech do kuchyně.

Součástí práce by měla být rešerše již existujících konkurenčních řešení a porovnání hlavních rozdílu a přínosu nově vzniklého systému.

3. Obecné principy

3.1. REST architektura

Jedná se o architekturu rozhraní pro distribuované prostředí (např. www), kterou popsal v roce 2000 Roy Fielding v rámci své disertační práce [1]. Umožňuje přistupovat k datům prostřednictvím standardních metod protokolu HTTP. Využívají se základní metody pod obecným označením CRUD (Creat - vytvořit, Retriev - získat, Update - změnit, Delete - odstranit). Označení metod v protokolu HTTP s popisem:

- **GET** – žádost o reprezentaci zdroje pomocí URL
- **POST** – slouží k odeslání uživatelských dat na server, používá se k vytváření nových objektů, záznamů apod.
- **PUT** – obdoba POST slouží pro změnu objektů, záznamů apod.
- **DELETE** – určeno pro odstranění objektů

REST je zkratkou slov Representational State Transfer, které popisují samotný princip architektury. Klient zažádá prostřednictvím definovaného URI o zdroj (data v nějakém formátu), následně server vrátí reprezentaci (Representation) dat, díky tomu se klient dostane do stavu (State). Při žádosti a obdržení jiného zdroje, přejde klient do jiného stavu, toto reprezentuje poslední ze slov (Transfer). Pro lepší pochopení lze uvést příklad: Chceme navštívit webové stránky zadáním adresu do prohlížeče (zažádáme o zdroj). Server nám vrátí reprezentaci webových stránek ve formátu HTML a tím se náš prohlížeč dostane do nějakého stavu. Při kliknutí na odkaz, který je umístěný na zobrazené stránce, zažádáme o nový zdroj a po přijetí dat se prohlížeč dostane do jiného stavu, než byl předtím.

Aby systém mohl být označován jako RESTful musí splňovat šest základních omezení definovaných v disertační práci Roye Fieldinga [1]. Jedná se o tato omezení:

- klient-server (client-server)
- bezstavovost (stateless)
- možnost využití vyrovnávací paměti (cacheable)
- kód na vyžádání (code on demand)
- vrstvený systém (layered system)
- jednotné rozhraní (uniform interface)

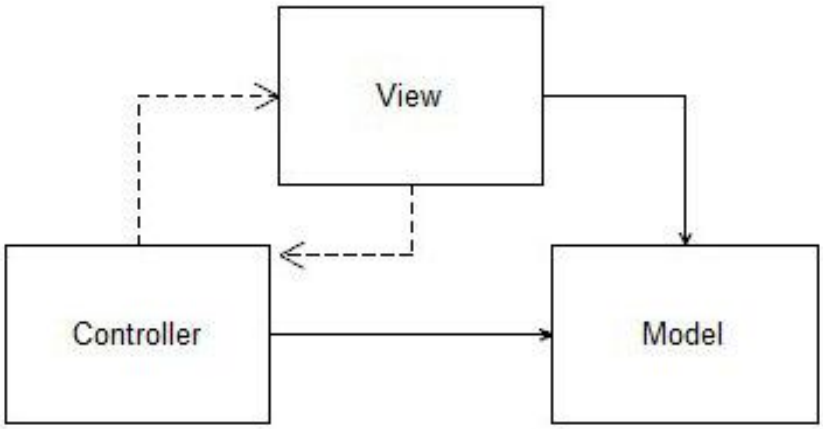
3.2. Architektura MVC

Při využití této architektury je aplikace rozdělena na tři části:

- **Model** – zastupuje datovou část aplikace
- **View** – zajišťuje zobrazení uživatelského rozhraní
- **Controller** – má na starost tok událostí v aplikaci

Samotný princip architektury je celkem prostý a nejlépe vysvětlitelný na příkladu. Představme si, že máme aplikaci pro expedici výrobků. Na obrazovce máme seznam expedovaných výrobku a dostupných výrobků k expedici. U každého dostupného výrobku máme tlačítko expedovat. Při jeho stisku obdrží controller oznámení o této události z objektu uživatelského rozhraní a aktualizuje model (přidá výrobek do seznamu expedovaných, aktualizuje výslednou cenu, atd.). Následně komponenta view použije aktualizovaný model a zobrazí aktualizovaný seznam výrobků k expedici.

Aplikace je tedy rozdělena do tří komponent a v případě nutnosti zásahu do jedné z nich (například změna designu pro zobrazení) je vliv na další dvě minimální.



Obrázek 1: Schéma architektury MVC

Z obrázku je vidět, že existují pouze dvě přímé vazby. Controller má přímý odkaz na model, aby mohl upravovat jeho data. A view má přímý odkaz, aby mohl zobrazovat data modelu. V některých variacích bývá ještě vazba mezi controllerem a view, někdy jednosměrná, popřípadě obousměrná.

3.3. Návrhové vzory

K návrhu a tvorbě systému byly použity i některé návrhové vzory. Návrhový vzor popisuje obecné řešení problému. Využívají se ve fázi návrhu a popisují jak daný problém řešit. Objektové orientované návrhové vzory popisují vztahy a interakce mezi třídami a objekty aniž by určovali konkrétní implementaci.

Návrhové vzory se dělí do tří základních skupin:

- **Creational Patterns** (vytvářející) – tyto vzory řeší problémy s vytvářením objektů v systému
- **Structural Patterns** (strukturální) – skupina návrhových vzorů, které jsou zaměřeny na možnosti uspořádání jednotlivých tříd nebo komponent v systému a tím ho zpřehlednit
- **Behavioral Patterns** (chování) – vzory zajímající se o chování systému, spolupráci mezi objekty atd.

3.3.1. Observer (pozorovatel)

Tento vzor spadá do kategorie Behavioral Patterns. Umožňuje sledovat změny objektu tak, že když objekt změní stav, na tuto změnu zareagují ostatní objekty. Přitom nedojde k přímé vazbě od sledovaného objektu k ostatním [2]. Platí zde vazba 1:N kde změna jedno objektu může ovlivnit libovolný počet závislých objektů. Teoreticky popisuje tento návrhový vzor RNDr. Ilja Kraval [2].

Návrhový vzor observer je přímo implementován v jazyce JAVA a proto se dá velmi snadno využít k tvorbě aplikace, která je postavena na architektuře MVC. K tomuto účelu bude také využit v aplikaci pokladna.

```
//vytvoření instance modelu
Model myModel      = new Model();
//vytvoření instance pohledu
View myView  = new View();

//zaregistruji pohled do seznamu observeru
myModel.addObserver(myView);
```

Použití vzoru Observer v jazyce JAVA

3.3.2. Singleton (jedináček)

Návrhový vzor spadající do kategorie Creational patterns, zabezpečující pouze jednu instanci objektu, která bude globálně viditelná ze všech částí aplikace [2]. Využívá se například pro objekt, který navazuje spojení s databází. Implementace tohoto vzoru je velmi jednoduchá. Třída, ze které se má objekt vytvořit, obsahuje soukromý konstruktor a statickou funkci. Ta při svém prvním zavolání vytvoří instanci třídy a uloží ji do statické proměnné, tuto jedinou instanci následně poskytuje dalším částem aplikace. Díky tomu je zajištěno, že je vždy pouze jedna instance objektu.

V navrhovaných aplikacích bude tento návrhový vzor využit pro objekt, který bude zajišťovat komunikaci s databází (webová aplikace) a pro objekt komunikující s API (aplikace pokladna).

```
//statická proměnná
private static Singleton instance;

 //soukromý konstruktor
 private Singleton() { }

//funkce pro vytvoření/získání instance
 public static Singleton getInstance() {
//pokud instance není vytvořená tak se vytvoří
       if (instance == null) {
             instance = new Singleton();
       }
       //vrátíme instanci
        return instance;
  }
```

Realizace návrhové vzoru singleton v jazyce JAVA

3.4. Základní popis platformy Android

Android je rozsáhlá open source platforma určená především pro mobilní zařízení. Operační systém je založený na jádru Linuxu a k programování se používá JAVA, která je upravena pro virtuální stroj Dalvik. Ten byl vyvíjen speciálně pro Android. Jako lokální datové úložiště se používá SQLite.

K programování pro tuto platformu je nezbytné nainstalovat následující sestavu software:

- **JDK** – (Java Development Kit) obsahuje soubor základních nástrojů pro vývoj aplikací pro platformu Java
- **Eclipse** – open source vývojové prostředí (IDE) pro jazyk JAVA (pomocí pluginů lze rozšířit i pro jiné jazyky)

- **ADK** (Android SDK) – nástroje pro vývoj aplikací k platformě Android
- **ADT Plugin** – (Android Development Tools) plugin pro vývojové prostředí Eclipse, umožňující vývoj aplikací Android

Každý program se skládá z tzv. aktivit, což je stavební kámen v této platformě. Každá aktivita odpovídá jedné obrazovce s uživatelským rozhraním pro interakci s uživatelem. Aplikace zpravidla obsahuje více aktivit, mezi kterými se uživatel pohybuje. Každá aktivita se může nacházet v jednom ze třech stavů:

- **Running** – aktivita je v popředí (na displeji) a dostává informace o uživatelském vstupu
- **Paused** - je částečně překrytá jinou aktivitou nebo že je nad ní jiná aktivita, která je však průhledná (při nedostatku paměti může být ukončena)
- **Stopped** – není vidět, jiná aktivita je ve stavu running, v případě nutnosti uvolnění zdrojů bude ukončena, lze znovu přepnout do stavu running
- **Killed** – aktivita byla ukončena ve stavu paused, nebo stopped

Vzhled aktivit je popsán v layout souboru (používá se formát xml), jsou zde definovány všechny prvky, které se na obrazovce zobrazí, a se kterými bude uživatel pracovat.

Asi nejdůležitějším souborem je AndroidManifest.xml. Jedná se o základní konfigurační soubor celé aplikace. Definují se zde jednotlivé aktivity, služby atd.

```
import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class MyActivity extends Activity
{
	/** Voláno když je aktivita poprvé vytvořena. */
	@Override
	public void onCreate(Bundle savedInstanceState)
	{
		super.onCreate(savedInstanceState);
		TextView tv = new TextView(this);
		tv.setText("Hello, world");
		setContentView(tv);
	}
}
```

Jednoduchý program pro platformu Android

4. Analýza, návrh řešení a výběr technologií

4.1. Existující řešení

Aktuálně je na našem trhu značném množství různých řešení dané problematiky od různých firem. Jelikož se jedná o komerční software, získané informace pochází z internetových prezentací jednotlivých výrobců. K porovnání byly zvoleny ty, které uvádí větší množství referencí a lze tedy předpokládat, že se jedná o úspěšné produkty. Všechny systémy musejí obsáhnout stejnou funkcionalitu. Proto zde nebudeme porovnávat jednotlivé funkce. Tato část bude spíše věnována koncepci samotných systémů a způsobu komunikace mezi jednotlivými moduly (aplikacemi).

Bude se jednat o nabídku následujících společností:

- A.W.I.S. Správa, systémy s.r.o. (Pokladní systém AWIS GASTRO)
- Agnis s.r.o. (Pokladna - restaurační software)
- ASW Systems, a.s. (Pokladní systém Septim)

4.1.1. A.W.I.S. Správa, systémy s.r.o. (Pokladní systém AWIS GASTRO)

Tento systém se skládá ze dvou aplikací. První s názvem Kasa slouží pro obsluhu restaurace a druhá s názvem Office je určena manažerům. Aplikace jsou nainstalovány na jednom PC, ke kterému je připojena tiskárna.

Nevýhody tohoto systému je především v instalaci obou aplikací na jednom počítači. Systém je sice možno ovládat vzdáleně, ale to znamená, že pokud by chtěl někdo z manažerů pracovat s aplikací Office mimo dobu kdy je restaurace otevřena, bylo by nutné nechat počítač zapnutý prakticky nonstop. Což zvyšuje nároky na energii a zároveň klesá životnost zařízení. Velkým mínusem je i absence mobilního číšníka pro jakoukoli platformu. Systém je nabízen včetně hardwaru a je vázán na platformu Windows, což opět může zvyšovat náklady.

4.1.2. Agnis s.r.o. (Pokladna - restaurační software)

Jedná se o systém, který může být tvořen jedním, popřípadě více počítači, kdy jeden musí být vždy určen jako hlavní. Na tomto počítači se vytváří sortiment restaurace, který je následně distribuován na ostatní stanice. Informace z jednotlivých stanic můžou být přenášeny prostřednictvím počítačové sítě, popřípadě na přenosných mediích. Pokud systém není propojen počítačovou sítí, lze editovat účet zákazníka pouze na stanici, kde byl vytvořen.

Tento systém nelze zpravovat vzdáleně skrz síť internet, ale pouze na lokální síti, kde je hlavní stanice umístěna, což je velké negativum.

V nabídce společnosti je již mobilní číšník. Je však dodáván na specializovaném hardware a v případě poruchy zařízení ho nelze jednoduše nahradit. Komunikace s hlavní stanici probíhá na lokální síti prostřednictvím wifi.

4.1.3. ASW Systems, a.s. (Pokladní systém Septim)

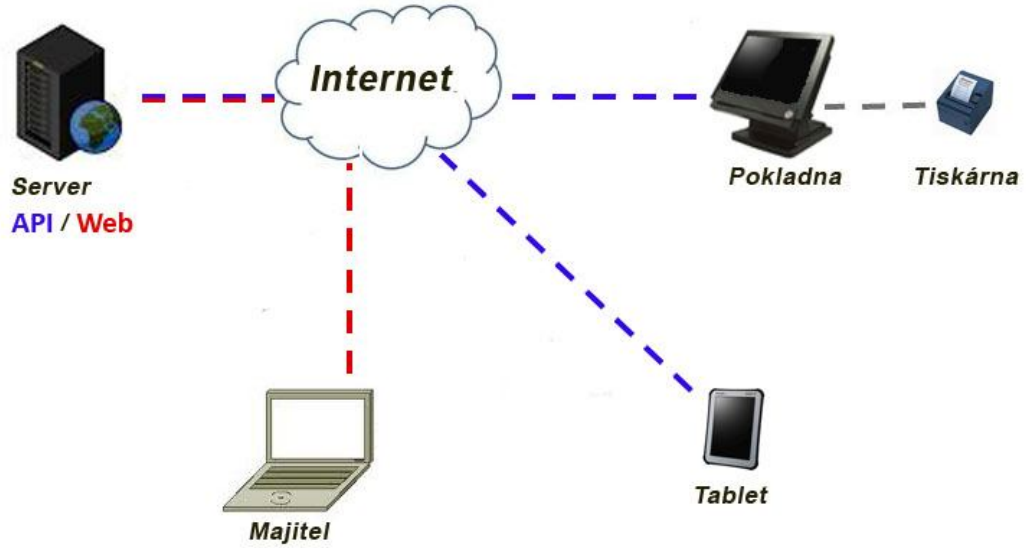
Systém je rozdělen opět na dvě části, kde jedna slouží pro personál a druhá pro vedení podniku. V tomto případě jsou nabízené různé moduly, které odpovídají specifičtějšímu zaměření podniku. Jedná se například o zaměření na fastfood, restauraci či jídelnu. U tohoto systému nelze provádět vzdálenou správu prostřednictvím sítě internet. Velikou výhodou je nezávislost na platformě.

Do systému, lze připojit mobilní pokladnu, kterou lze využívat jako mobilního číšníka. Jedná se však o velmi nákladný hardware.

4.2. Návrh řešení

Při návrhu řešení byl kladen velký důraz na nezávislost na platformě. Toho bylo docíleno u aplikace provozního (zvoleno webové řešení) a pokladny (zvolen programovací jazyk JAVA). U mobilního číšníka již nezávislost dodržet nelze a je nutné vybrat určitou platformu, pro kterou bude aplikace vyvinuta. Díky tomu, že platforma Android je plně otevřený systém a zároveň je dostupné široké spektrum zařízení, byla zvolena právě ona.

Po seznámení s nabídkou již existujících řešeních byla odhalena velká slabina těchto softwarů. Jedná se o vzdálenou správu sortimentu a přístup k informacím o chodu podniku. Většina systému toto neumožňuje, a pokud ano je nutné mít zapnutý hlavní počítač systému, který je umístěn v podniku. Z toho důvodu bylo zvoleno řešení, kde hlavní počítač (server) bude umístěn mimo podnik v rámci sítě internet. Vedoucí podniku budou moci provádět manažerské operace pomocí webových stránek a jednotlivé aplikace s ním budou komunikovat pomocí API.



Obrázek 2: Schéma komunikace systému prostřednictvím protokolu HTTP

4.2.1. Webová aplikace

Při vývoji této aplikace je důležité klást důraz na snadnou rozšiřitelnost a úpravy. Každý podnik má své zažité rutiny a sleduje jiné ukazatele běhu podniku. Proto tato aplikace bude psána pomocí OOP a modulovým principem. Díky tomu úpravy na míru jednotlivým zákazníkům ušetří čas i náklady.

Aby aplikace splňovala veškeré potřeby provozního, musí obsáhnout velký rozsah témat. Díky zvolenému modulovému principu lze v počátcích vývoje implementovat pouze základní oblasti, bez kterých se podnik neobejde, a následně aplikaci rozšiřovat. V rámci diplomové práce budou implementovány tyto okruhy tvořeny samostatnými moduly:

- **Zboží** (editace nabízeného sortimentu v podniku)
- **Kategorie** (editace kategorií sortimentu)
- **Nastavení** (editace jednotlivých stolů podniku, uživatelů)
- **Objednávky** (přehled objednávek a tržeb)

Součástí aplikace bude ještě speciální modul. Bude se jednat o API rozhraní, prostřednictvím kterého budou získávat a zpětně zasílat data ostatní aplikace (pokladna, mobilní číšník).

4.2.2. Pokladna

Mělo by se jednat o desktopovou aplikaci, která bude instalovaná na libovolném PC. Pomocí této aplikace bude moci obsluha plně obsloužit hosta od jeho příchodu až po konečné placení. Jednou z hlavních úloh bude tisk účtenek a lístků s objednanými pokrmy, které se mají v kuchyni připravit. Toto bude probíhat na pozadí aplikace, kde bude periodickými dotazy na server hlídáno, zda nemá dojít k vytištění nějakého lístku.

Prostřednictvím aplikace bude personál k jednotlivým stolům vytvářet účty, kterých bude neomezené množství, a do těchto účtů budou přidávány jednotlivé položky, které si zákazník objedná. Jednotlivé účty bude potřeba také slučovat, popřípadě umožnit personálu jednotlivé účty pojmenovat podle svého. Součástí aplikace bude obrazovka tzv. „fronta nápojů“ bude se jednat o objednané položky, které si zákazníci objednali, ale ještě jim nebyly doneseny personálem dodány. Tuto obrazovku může využít např. barman a připravovat objednané nápoje s předstihem a číšník je už následně pouze dodá zákazníkovi na stůl. Někdy se však můžou objednané položky rovnou dodat zákazníkovi. A v tuto chvíli je využití fronty nápojů nežádoucí, proto aplikace bude muset umět položku vložit jak do fronty, tak přímo mezi dodané položky.

Nedílnou částí musí být také možnost zaplacení útraty. Zde je důležitá možnost zaplacení pouze určitých položek z lístku. Po potvrzení bude automaticky proveden tisk účtenky a odečtení uhrazených položek, popřípadě smazání celého účtu.

4.2.3. Mobilní číšník

Pro tuto aplikaci byla vybrána platforma Android a bude tedy určená pro zařízení s tímto operačním systémem. Aby tato aplikace měla smysl, musí umět číšníkovi nabídnout podobnou funkcionalitu jako velká pokladna. Číšníkovi tak odpadá nutnost neustále odbíhat a zadávat jednotlivé objednávky do systému. Komunikace bude probíhat prostřednictvím API, které bude umístěno na serveru. K tomuto API se každý číšník bude přihlašovat prostřednictvím svého jména a hesla. Díky tomu, bude moci vedoucí evidovat, kdo z personálu byl jak aktivní.

Aplikace tedy umožní, přepínání účtů u jednotlivých stolů. A přidávat do těchto účtů jednotlivé položky a to jak do fronty nápojů, tak do dodaných položek.

5. Realizace

Tato kapitola se bude týkat samotné realizace jednotlivých aplikací. Vzhledem k tomu, že celá práce se skládá ze tří aplikací a komunikačního API bude kapitola rozdělena na čtyři části, která se bude vždy věnovat jedné z aplikací.

Pro potřeby testování byla webová aplikace (včetně API) uvedena do provozu na adrese http://dp.insert.cz, kde s ní mohou ostatní aplikace komunikovat.

5.1. Realizace webové aplikace

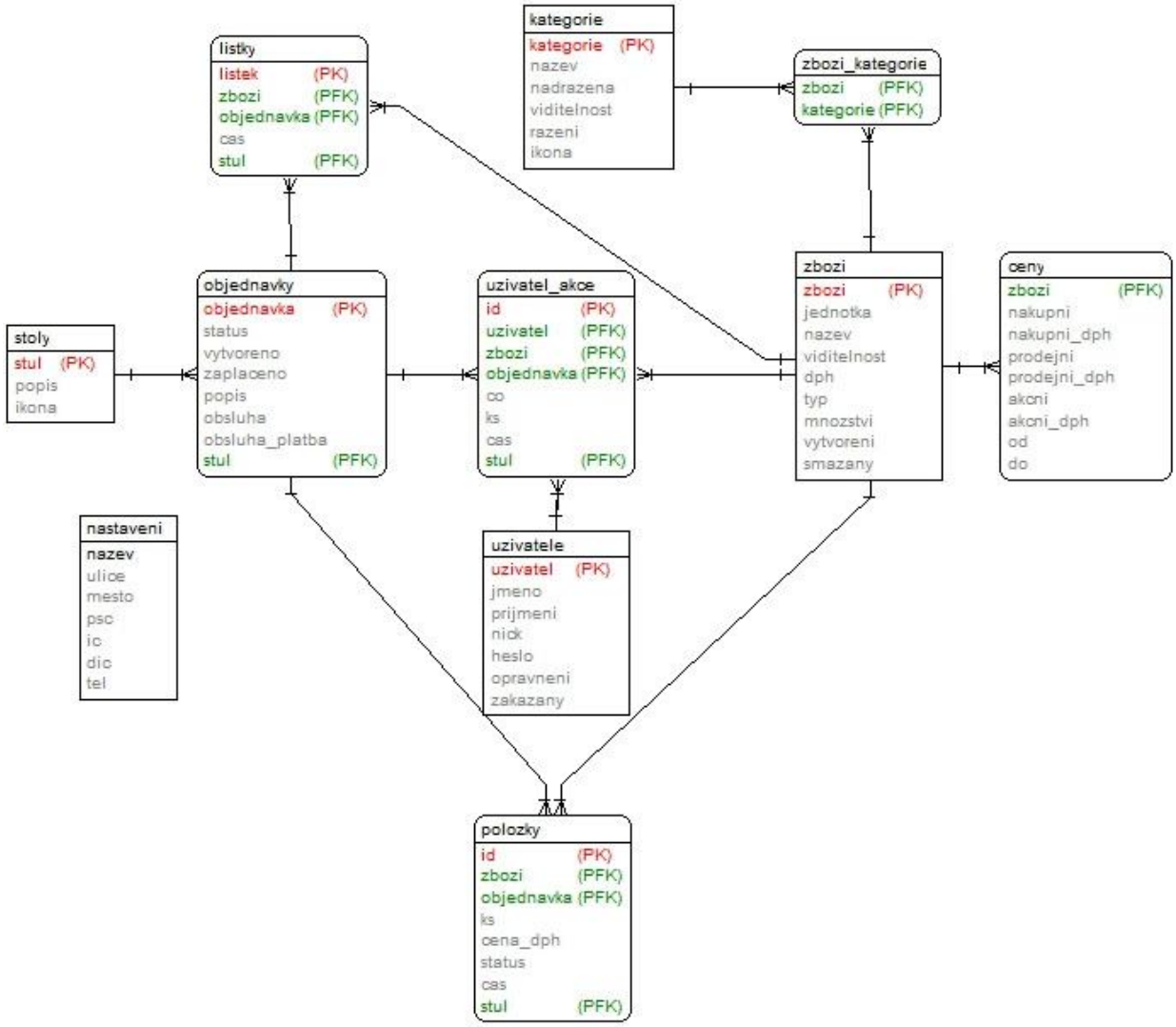
K vytvoření aplikace, byly použity tyto technologie:

- **HTML, CSS, JavaScript** – standardní technologie pro tvorbu webových stránek
- **PHP** – skriptovací programovací jazyk určený především pro programování webových aplikací (použit ve verzi 5.4.3)
- **Nette** - framework pro tvorbu webových aplikací v PHP 5
- **AJAX** - obecné označení technologií, které slouží pro vývoj interaktivních webových stránek, které mění obsah (popř. část svého obsahu) bez nutnosti jejich opětovného načítání
- **jQuery** – javascriptový framework

Jednotlivé moduly jsou psány objektově orientovaným způsobem. Každý modul obsahuje inicializační soubor *main.php*, který inicializuje jednotlivé objekty a zobrazí základní obrazovku modulu.

5.1.1. Databáze

Pro požadavky systému byl zvolen relační databázový systém MySQL. Tento systém je multiplatformní a komunikace s ním probíhá prostřednictvím jazyka SQL. Celkem se vytvořená databáze skládá z jedenácti tabulek. Kde deset z nich je vzájemně provázáno a jedna je naprosto samostatná. Tato samostatná tabulka slouží pro uložení kontaktních údajů podniku.



Obrázek 3: ER diagram databáze

5.1.2. Adresářová struktura aplikace

Jelikož je aplikace psána modulově je důležité dodržovat adresářovou strukturu.

- **API/ -** adresář s API rozhraním
- **API/Class -** adresář s třídami, které využívá pouze API
- **config/** - adresář konfiguračních souborů
- **css/** - adresář s css stylem a souborem funkcí v jazyce JavaScript
- **css/image/** – adresář obsahující obrázky vzhledu
- **js/** - adresář obsahující jQuery framework
- **moduly/** - adresář obsahující jednotlivé moduly
- **moduly/kategorie** – adresář modulu pro práci s kategoriemi

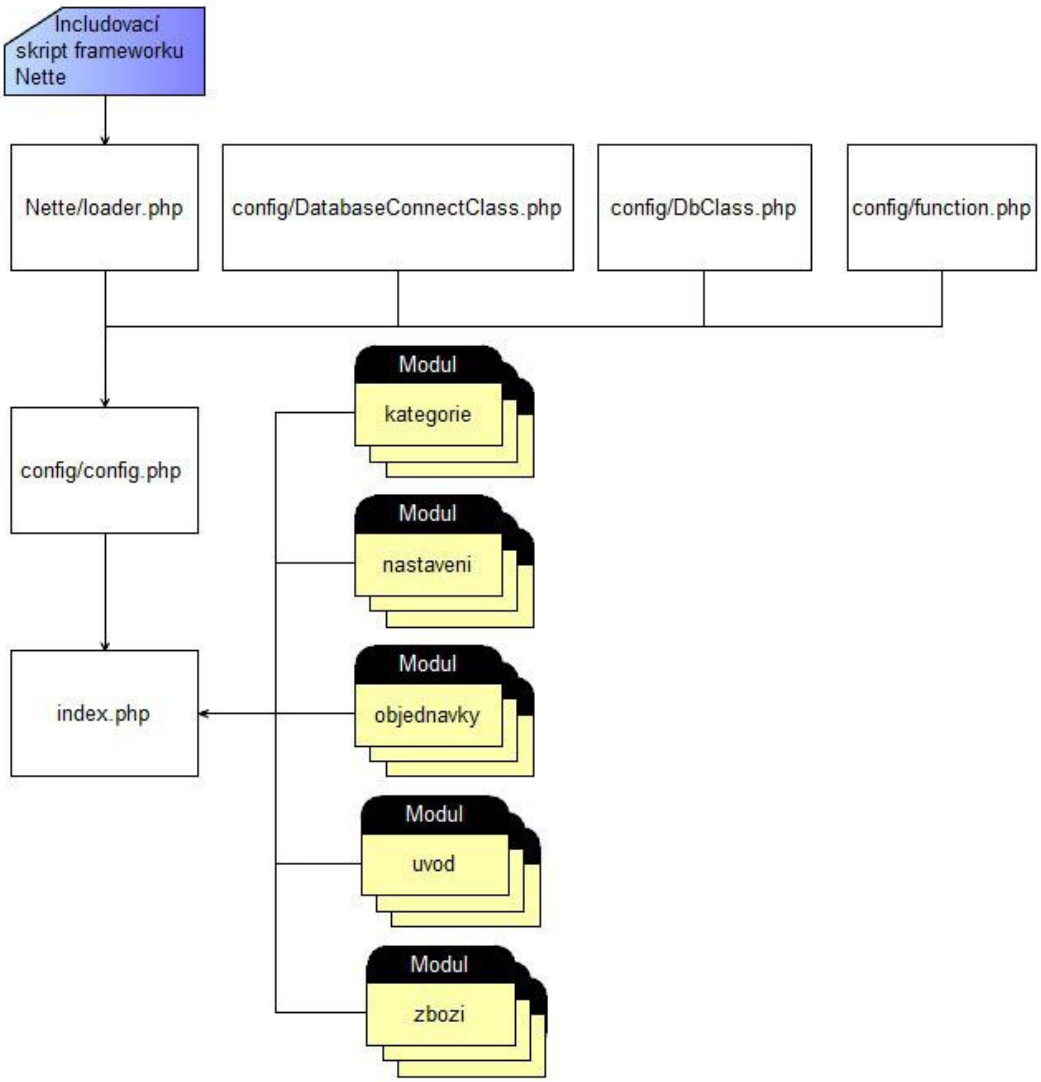
- **moduly/kategorie/ajax –** adresář se skripty volanými skrze AJAX modulu kategorie
- **moduly/kategorie/class** – adresář obsahující jednotlivé třídy modulu kategorie
- **moduly/ nastaveni** – adresář modulu pro nastavení
- **moduly/nastaveni/ajax –** adresář se skripty volanými skrze AJAX modulu nastavení
- **moduly/nastaveni/class** – adresář obsahující jednotlivé třídy modulu nastavení
- **moduly/objednavky** – adresář modulu pro zobrazení objednávek a aktivity personálu
- **moduly/objednavky/ajax –** adresář se skripty volanými skrze AJAX modulu objednávky
- **moduly/objednavky/class** – adresář obsahující jednotlivé třídy modulu objednávky
- **moduly/uvod** – adresář se základním modulem
- **moduly/zbozi** – adresář modulu pro správu zboží
- **moduly/zbozi/ajax** – adresář se skripty volanými skrze AJAX modulu zboží
- **moduly/zbozi/class** – adresář obsahující jednotlivé třídy modulu zboží
- **Nette/ -** adresář s frameworkem Nette
- **picture/ -** adresář s obrázky

5.1.3. Souborová struktura aplikace

Hlavním souborem, který se vždy vykoná je *index.php*, který se nachází v kořenovém adresáři. Stará se o načítání všech souborů, které jsou aktuálně potřebné. Jako jediný obsahuje HTML hlavičku. Další důležité soubory s umístěním a popisem:

- **index.php** – hlavní soubor načítající potřebné skripty
- **config/config.php** – konfigurační skript, nastavuje aktuální modul a vkládá základní třídy
- **config/DatabaseConnectClass.php** – třída, která vytváří pouze jednu instanci připojeni k databázi a v případě opakovaného volaní vrací vždy tento jeden objekt (využit návrhový vzor Singleton)
- **config/DbClass.php** – třída obsahující vlastní funkce pro sestavování SQL dotazů a komunikaci s databází
- **config/function.php -** soubor obsahující vlastní funkce

- **config/JednotkyClass.php** – třída pro jednoduché převody mezi používanými jednotkami
- **css/function.js** – soubor obsahující JavaScriptové funkce
- **css/style.css** – soubor s css stylem pro vzhled systému



Obrázek 4: Souborové schéma aplikace

5.1.4. Modul kategorie

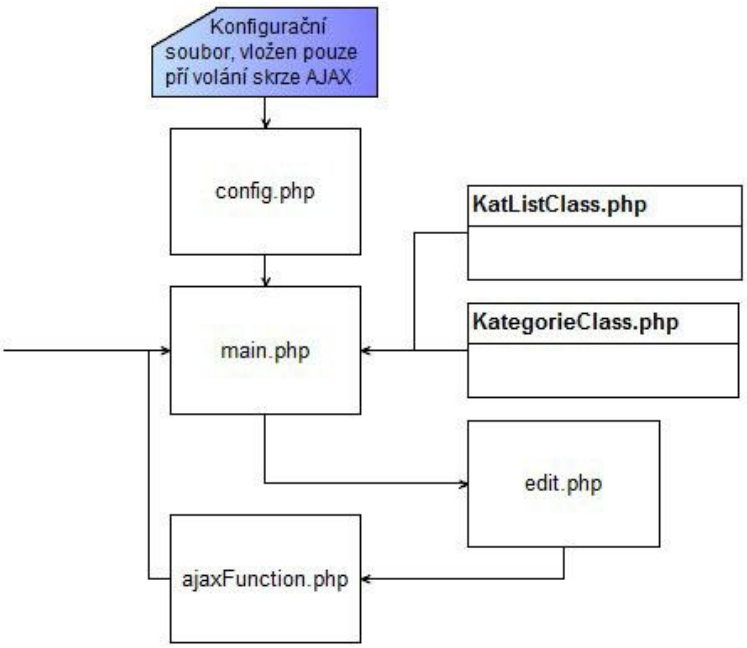
Tento modul slouží pro vytváření a editaci jednotlivých kategorií produktů. Pro zvýšení přehlednosti umožňuje uživateli vytvářet, z jednotlivých kategorií, stromovou strukturu. U každé kategorie jsou evidovány tyto údaje:

- **Název** (jméno kategorie)
- **Nadřazená** (kategorie, pod kterou bude tato umístěna)

- **Ikona** (obrázek, který bude zobrazen spolu s názvem kategorie)
- **Viditelnost** (zda bude kategorie zobrazena či ne)

Samotný modul je tvořen dvěma třídami. První (*KatListClass.php*) slouží pro listování stromem kategorií, kde každá kategorie je objektem třídy (*KategorieClass.php*). V každé instanci této třídy jsou definovány všechny údaje konkrétní kategorie a prostřednictvím nabízených metod lze tyto údaje měnit a ukládat.

Modul obsahuje dále soubor *edit.php*, který se volá v jQuery dialogu a slouží pro editaci kategorie. Po uzavření dialogu, dojde k opětovnému zavolání hlavního skriptu modulu (*main.php*) a novému načtení kategorií, k tomuto volání dochází skrze AJAX. Provedené změny z editace kategorie se uloží pomocí skriptu v souboru *ajaxFunction.php* umístěném v adresáři ajax.



Obrázek 5: Souborové schéma modulu kategorie

5.1.5. Modul nastavení

Modul umožňující nastavení některých parametrů podniku. Konkrétně se jedná o možnost vytvářet libovolný počet stolů, uživatelů a vyplnit kontaktní údaje o restauraci.

Editace stolů:

Vytvořené stoly se zobrazují v aplikacích, které využívá personál. Jednotlivé účty zákazníků jsou vždy přiřazeny k některému ze stolů. U každého stolu jsou definovány tyto údaje:

- **Popis** (Vlastní pojmenování stolu pro lepší přehlednost)
- **Typ** (jedná se o výběr, zda stůl bude reprezentován kulatou ikonou nebo hranatou)

Editace uživatelů:

V systému fungují dvě úrovně uživatelů, které se přidělují podle toho, zda se jedná o personál, který obsluhuje pouze zákazníky, nebo zda bude mít přístup do webové aplikace, která je určena vedoucím pracovníkům. Údaje, které jsou definovány u každého uživatele:

- **Jméno** (jméno uživatele)
- **Příjmení** (příjmení uživatele)
- **Nick** (název uživatele, pomocí kterého se přihlašuje do systému)
- **Heslo** (heslo, pomocí kterého se uživatel přihlašuje)
- **Aktivní** (volba ano/ne zda je uživatel aktivní a může se přihlásit do systému)
- **Oprávnění** (volba úrovně uživatele vedení/personál)

Kontaktní údaje:

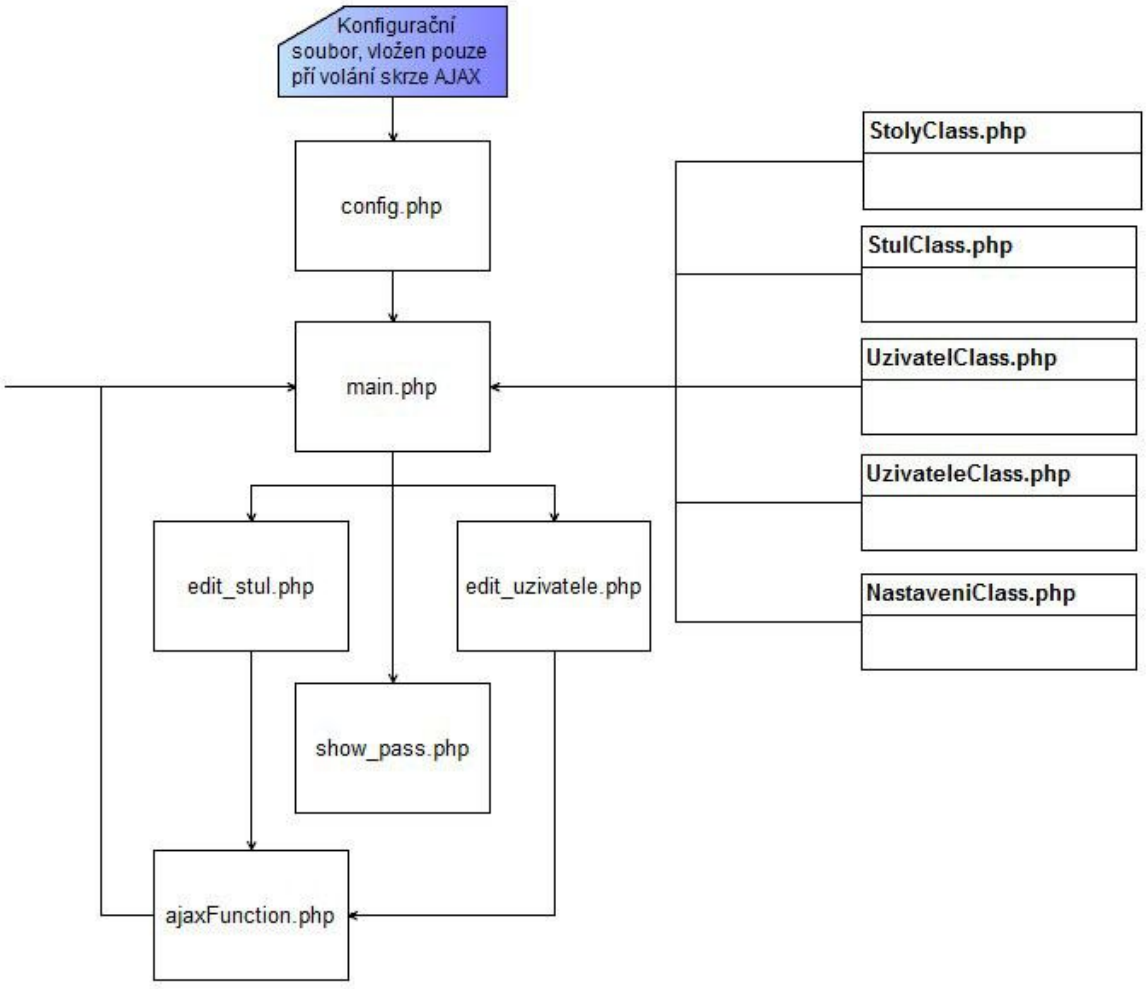
Vyplněné kontaktní údaje se využívají při tiku účtenek, které se předávají zákazníkovi po zaplacení. Jedná se o tyto údaje:

- **Název** (název podniku)
- **Ulice** (ulice kde se podnik nachází)
- **Město** (město kde se podnik nachází)
- **PSČ** (poštovní směrovací číslo města, kde se podnik nachází)
- **IČ** (identifikační číslo provozovatele)
- **DIČ** (daňové identifikační číslo provozovatele)
- **Tel** (telefonní číslo do podniku)

Modul pro nastavení se skládá celkem z pěti tříd. Dvě slouží pro operace se stoly, dvě pro uživatele a jedna pro nastavení kontaktních údajů.

- *StulClass.php* – třída definující vlastnosti konkrétního stolu
- *StolyClass.php* – třída pro listování stolů
- *UzivatelClass.php* – třída definující vlastnosti konkrétního uživatele
- *UzivateleClass.php* – třída pro listování uživatelů
- *NastaveniClass.php* – třída sloužící pro nastavení kontaktních údajů

Modul obsahuje ještě další skripty, které slouží k editacím jednotlivých částí. K přidání stolu, nebo jeho editaci slouží skript *edit_stul.php,* kde se vyplňují jednotlivé parametry. Pro úpravy uživatele, nebo vytvoření nového je skript *edit_uzivatele.php.* Oba tyto skripty se otevírají v dialogovém okně a k uložení změn dojde ve skriptu *ajaxFunction.php*. V případě zapomenutí hesla ho lze zobrazit přímo na výpisu seznamu uživatelů pomocí skriptu *show_pass.php*, který se také otevře v dialogovém okně. Formulář pro vyplnění kontaktních údajů je umístěn v hlavním skriptu modulu *main.php*.



Obrázek 6: Souborové schéma modulu nastavení

5.1.6. Modul objednávky

Jedná se o modul, který slouží pro zobrazení přehledu objednávek v podniku a orientační přehled o aktivitě jednotlivého personálu.

Přehled objednávek:

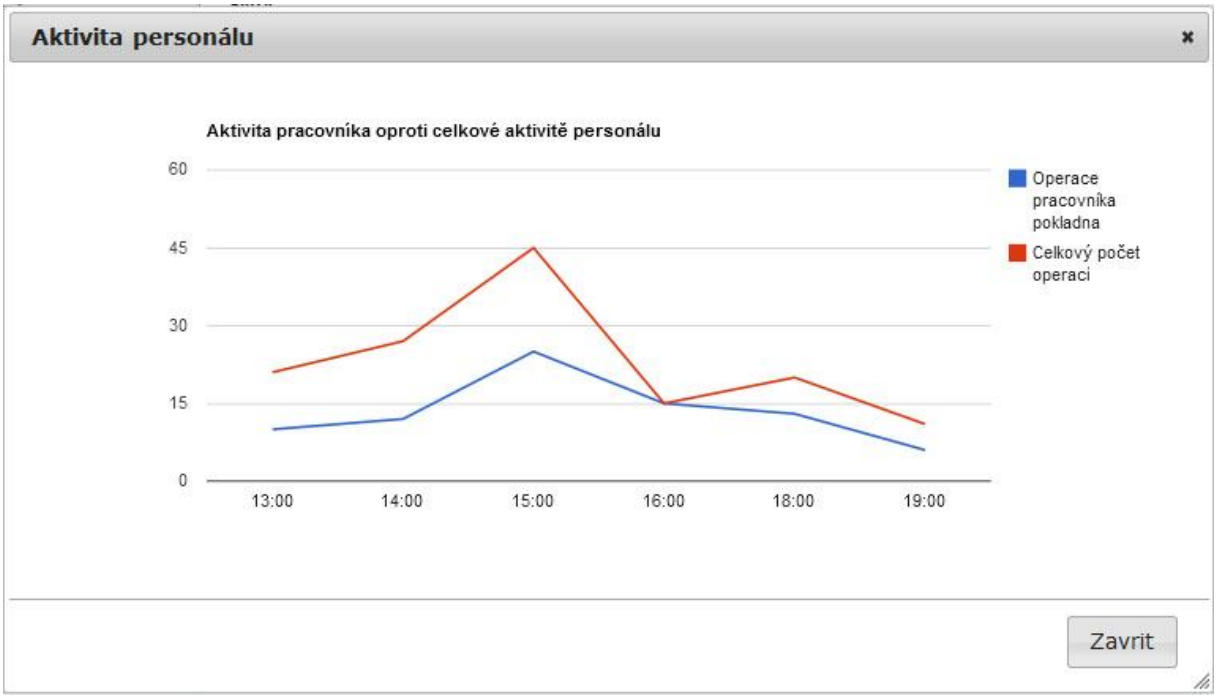
Pro výběr konkrétních objednávek je zde možnost filtrovat podle data vytvoření (od, do) a podle stolu, ke kterému byly objednávky vytvořeny. Objednávky jsou rozděleny na dvě části podle toho, zda je již objednávka uzavřena (všechny položky jsou zákazníkem zaplaceny) anebo zda v ní jsou ještě položky, které na uhrazení čekají. V seznamu objednávek se každé objednávky zobrazují tyto údaje:

- **Id** (id objednávky)
- **Stůl** (stůl, u kterého zákazník seděl a ke kterému byla objednávka vytvořena)
- **Vytvoření** (datum a čas vytvoření objednávky)
- **Zaplacení** (datum a čas zaplacení objednávky, pouze v případě, že je objednávka uzavřena)
- **Cena** (celková cena objednávky)
- **Vytvořil** (nick zaměstnance, který objednávku vytvořil)
- **Uzavřel** (nick zaměstnance, který objednávku uzavřel, pouze v případě, že je objednávka uzavřena

U každé objednávky je možnost nechat si zobrazit detailní výpis. Jedná se o zobrazení objednávky včetně položek, které si zákazník objednal.

Aktivita personálu:

Zobrazit aktivitu lze ke konkrétním datům od, do. Poté se zobrazí seznam pracovníků, kteří mají záznam o provedené operaci (přijmutí, úprava objednávky zákazníka). Ke každému zákazníkovi je možné nechat si zobrazit graf, kde je vidět počet jeho operací a celkový počet všech operací v podniku ve vybraném časovém úseku.



Obrázek 7: Graf aktivity personálu

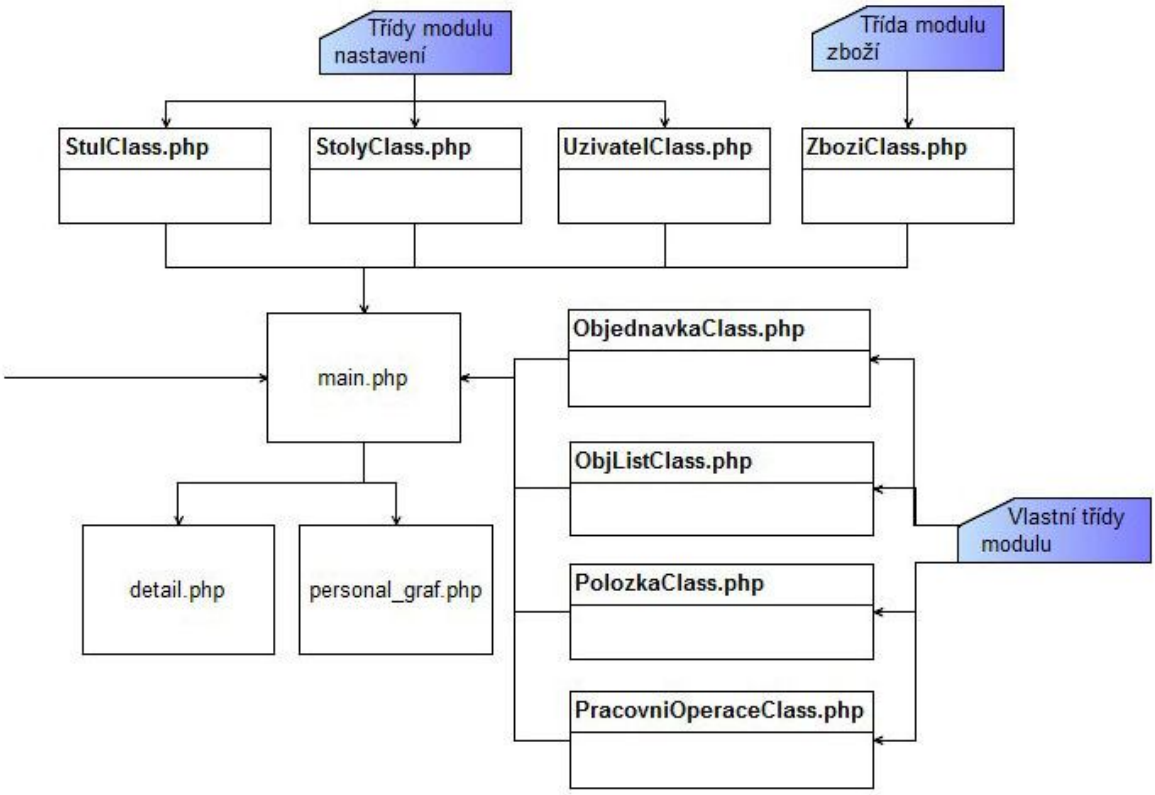
Celkem se modul skládá ze čtyř vlastních tříd a čtyř, které využívá z jiných modulů. Vlastní třídy modulu:

- *ObjednavkaClass.php* – třída definující konkrétní objednávku
- *ObjListClass.php* – třída, která slouží pro listování objednávek
- *PolozkaClass.php* – třída, která definuje konkrétní položku v objednávce
- *PracovniOperaceClass.php* – třída pro zobrazení aktivity personálu

Využívané třídy z jiných modulů:

- *StulClass.php* – třída modulu nastavení (složí pro vylistování stolů do filtru)
- *StolyClass.php* – třída modulu nastavení (využitá k vylistování stolů do filtru a zobrazení konkrétního stolu u objednávky)
- *UzivatelClass.php* – třída modulu nastavení (použita pro zobrazení konkrétního pracovníka u objednávky)
- *ZboziClass.php* – třída modulu zboží (slouží pro vytvoření seznamu objednaných položek v objednávce)

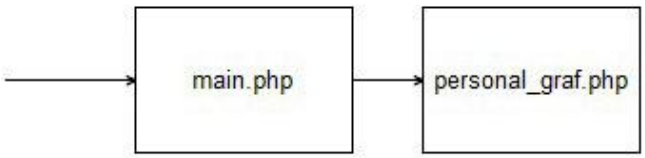
K zobrazení detailu objednávky slouží skript *detail.php* a pro zobrazení grafu aktivity pracovníka se využívá skript *personal_graf.php*. Oba tyto skripty se otevírají v dialogovém okně. K vykreslení grafu se používá služba společnosti Google s názvem Google Chart API.



Obrázek 8: Souborové schéma modulu objednávky

5.1.7. Modul úvod

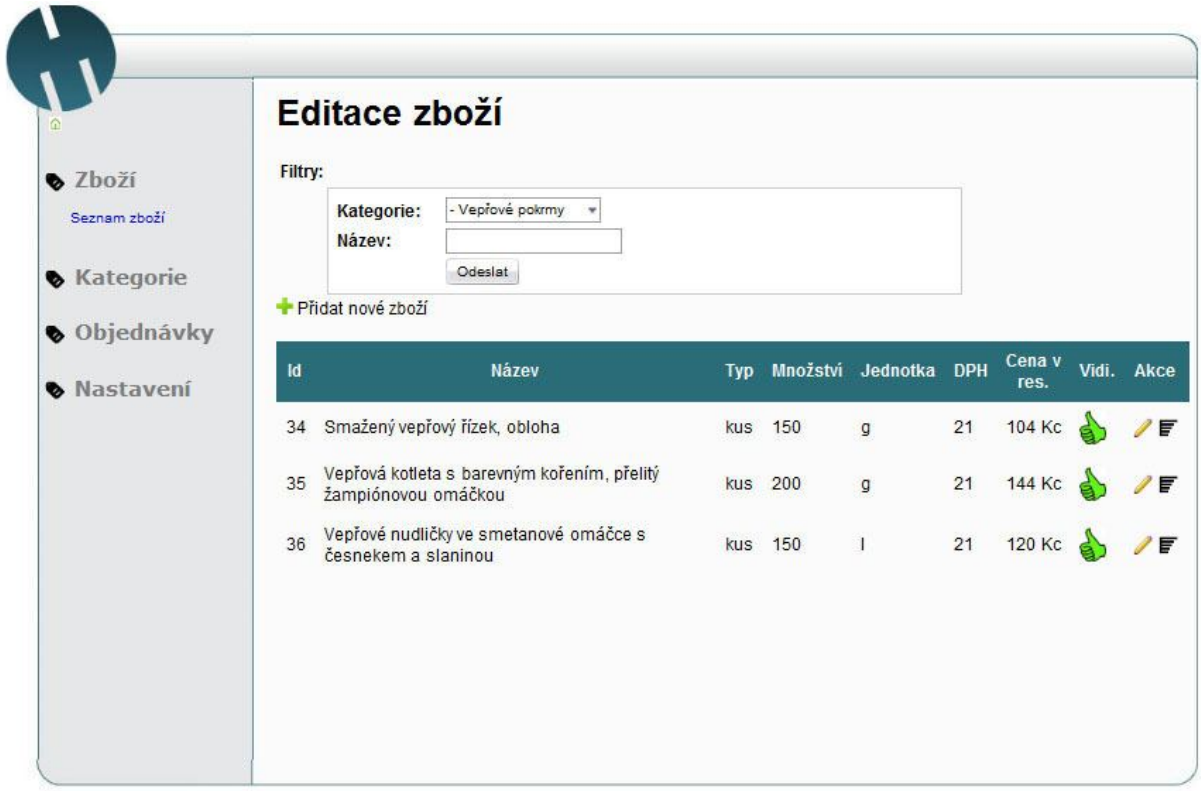
Tento modul slouží pouze pro zobrazení úvodní obrazovky. Skládá se pouze z hlavního skriptu *main.php* a *info.php*. Tento skript vypíše pevně stanovený řetězec.



Obrázek 9: Souborové schéma modulu úvod

5.1.8. Modul zboží

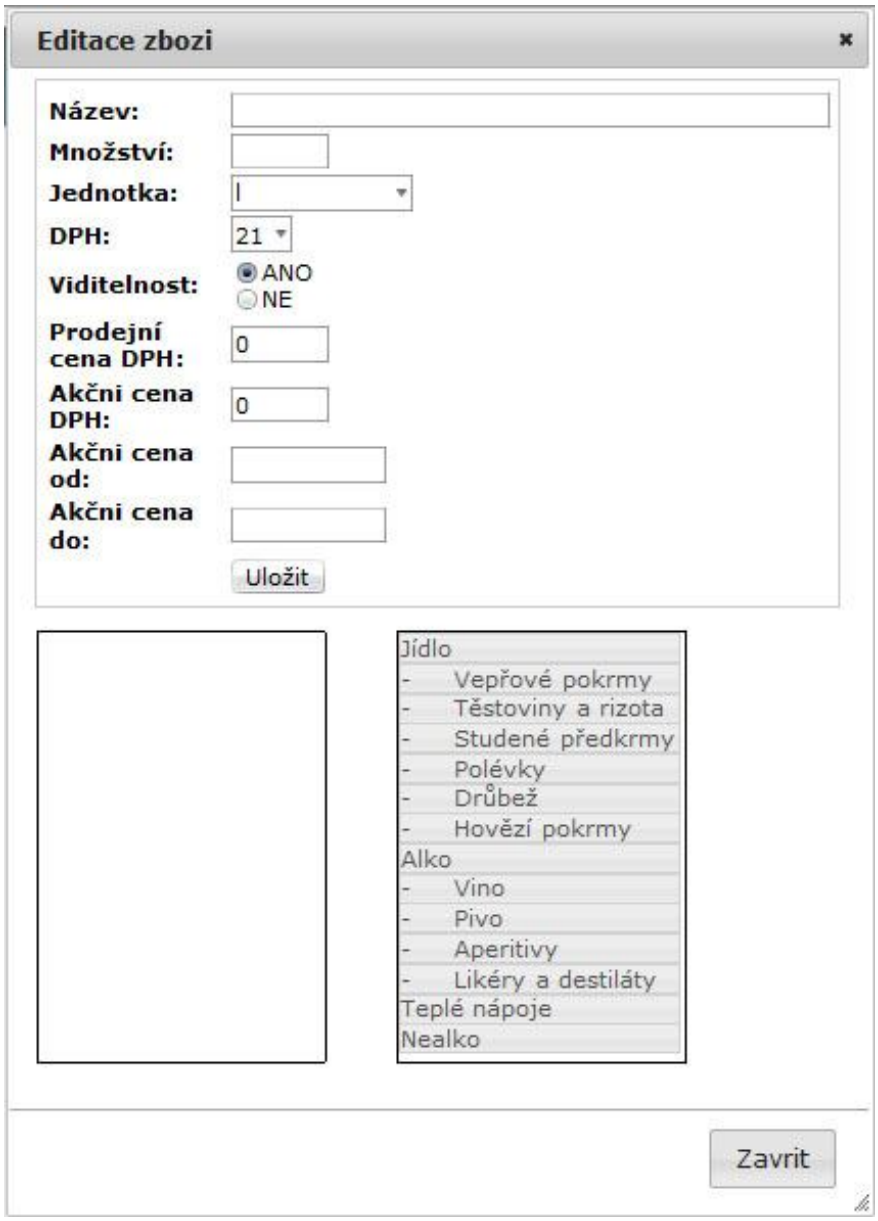
Pomocí tohoto modulu lze prohlížet aktuální nabídku podniku a provádět změny v sortimentu. Zobrazení lze filtrovat podle dvou kritérií, a sice podle kategorie produktů nebo přímo podle názvu.



Obrázek 10: Modul zboží

U každého produktu jsou evidovány tyto údaje:

- **Název** (název produktu)
- **Množství** (prodávané množství)
- **Jednotka** (množstevní jednotka)
- **DPH** (výše DPH)
- **Viditelnost** (nastavení zda je produkt zobrazen v nabídce)
- **Prodejní cena** (cena produktu)
- **Akční cena** (akční cena produktu)
- **Akční cena od** (datum od kdy platí akční cena)
- **Akční cena do** (datum do kdy platí akční cena)
- **Kategorie** (kategorie, do kterých produkt patří)



Obrázek 11: Editační okno produktu

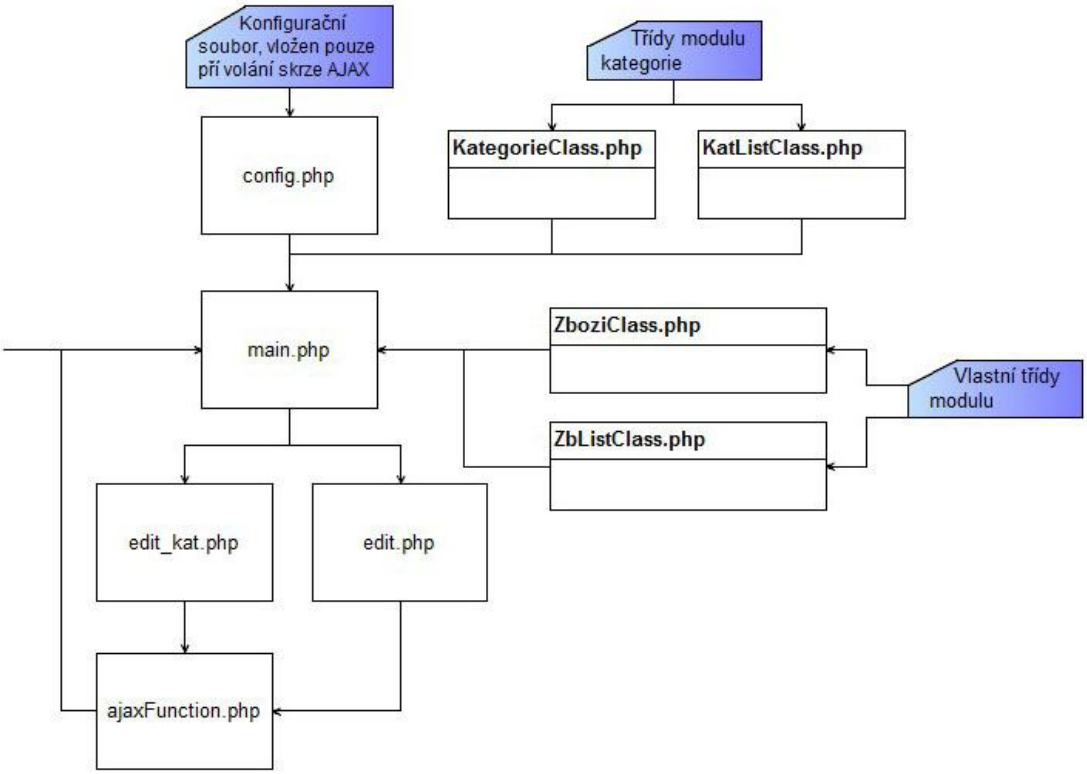
Modul zboží je úzce provázán s modulem kategorie, jelikož každý produkt musí být zařazen alespoň do jedné kategorie. Proto jsou využívány i třídy tohoto modulu. Vlastní třídy modulu:

- *ZboziClass.php* – třída definující konkrétní zboží
- *ZbListClass.php* – třída pro listování zbožím

Využívané třídy modulu kategorie:

- *KategorieClass.php*
- *KatListClass.php*

K editaci a vkládání nových produktů slouží skript *edit.php,* kde se editují všechny vlastnosti kromě kategorií, do kterých produkt patří. K tomu slouží skript *edit_kat.php*, kde se pomocí JqueryUI Sortable přetahují jednotlivé kategorie. Oba editační skripty se otevírají v dialogovém okně. Po odeslání dat se provedené změny uloží pomocí skriptu *ajaxFunction.php*. A následně dojde k opětovnému zavolání hlavního skriptu modulu *main.php* a novému načtení seznamu produktů.



Obrázek 12: Souborové schéma modulu zboží

5.2. API

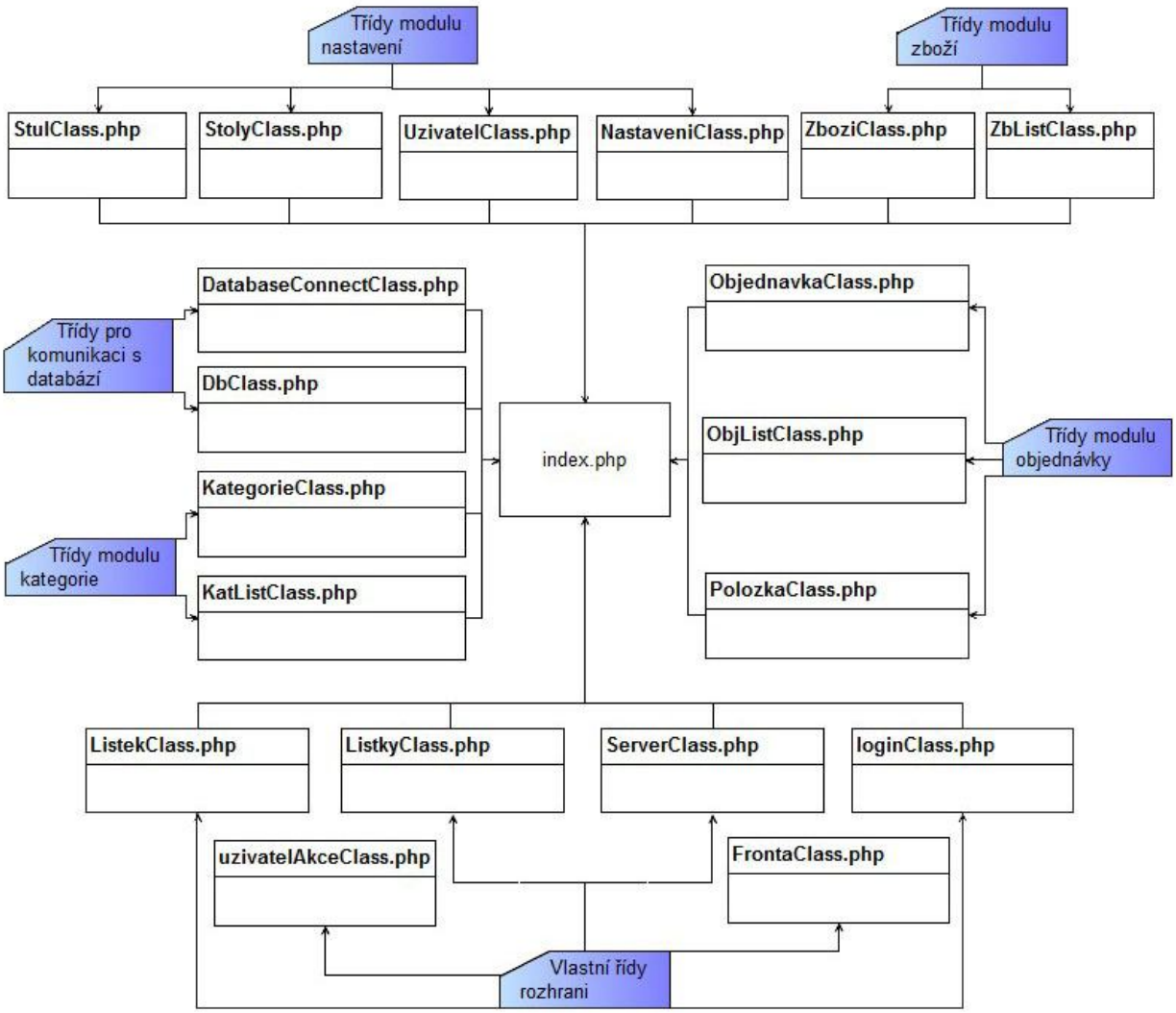
Toto rozhraní umožňuje komunikovat aplikacím, které používá personál (pokladna, mobilní číšník), se serverem kde je umístěna databáze webové aplikace, ve které se vytváří sortiment restaurace, nastavují stoly atd. Jelikož vytvořené API budou využívat dvě aplikace na různých platformách, je výhodnější využít architekturu, které je orientovaná datově a nikoli procedurálně. Z tohoto důvodu byla zvolena architektura REST, která zároveň umožňuje pro vzájemnou komunikaci využít různé formáty dat (ATOM/RSS, JSON, XML). Vzhledem k charakteru přenášených dat (maximálně dvourozměrné pole) byl zvolen formát JSON, který je lehce čitelný a nezávislý na platformě.

5.2.1. Struktura

Vytvořené API je implementováno jako samostatný speciální modul webové aplikace, protože využívá zdroje této aplikace (databáze) a jednotlivé třídy ostatních modulů. Přesto obsahuje i několik vlastních tříd. Umístěn je ve speciálním adresáři s názvem API, kde je umístěn i konfigurační soubor *.htaccess* pro upravení chování webového serveru. Přesněji je v něm nastaveno přesměrování všech požadavků vedoucí do složky API do souboru index.php, který je hlavním skriptem rozhraní. Vkládají s v něm všechny využívané třídy ať vlastní či z ostatních modulů. Následně se vytvoří instance třídy *ServerClass.php*, které je obsluhující třídou rozhraní. Po vytvoření objektu této třídy je zavolána funkce serve(), která slouží jako jakýsi rozcestník. Dle zadaného URI volá konkrétní funkci, která vyřídí požadavek a vrátí výsledná data, která se pošlou zpět aplikaci.

Vlastní třídy API:

- *FrontaClass.php* – třída, obsluhující požadavek na zobrazení položek ve frontě
- *ListekClass.php* – třída definující konkrétní lístek do kuchyně
- *ListkyClass.php* – třída pro listování lístky do kuchyně
- *loginClass.php* – třída sloužící pro přihlašování k API
- *ServerClass.php* – hlavní třída API
- *uzivatelAkceClass.php* – třída sloužící k zaznamenávání aktivity personálu



Obrázek 13: Souborové schéma API

5.2.2. Komunikace API

Jedná se o neveřejné API a je tedy důležité, aby s ním nemohl pracovat každý, ale pouze schválení uživatelé, proto obsahuje autentizační mechanizmus. Aktuálně se využívá základní HTTP autentizace. Kde se při každém requestu (požadavku) v http hlavičkách posílá uživatelské jméno a heslo, tyto údaje se porovnají s daty vytvořených uživatelů z webové aplikace a jejich oprávnění v případě, že nenalezení shody se zašle hlavička 401 Access denied. Pokud dojde k úspěšnému přihlášení, jsou aplikaci k dispozici následující URI:

/stoly

URI svázané s REST metodu typu: **GET**.

Určení: Po zavolání vrátí seznam stolů v podniku.

Odesílané parametry:

- **Id** – id stolu v systému
- **Popis** – popis stolu definovaný ve webové aplikaci
- **Ikona** – název souboru podle vybraného typu (hranatý, kulatý)
- **Obsazený** – informace o tom zda je nějaký aktivní účet svázaný s tímto stolem

/kategorie

URI svázané s REST metodu typu: **GET**.

Určení: Slouží k vylistování kategorií, které jsou vytvořené v systému a mají nastavenou viditelnost k zobrazení v podniku.

Odesílané parametry:

- **Id** – id kategorie v systému
- **Ikona** – obrázek kategorie, sloužící ke snazšímu identifikování kategorie v přehledu, který je pro přenos kódovaný pomocí base64
- **Name** – název kategorie

/kategorie/XX

URI svázané s REST metodu typu: **GET**.

Určení: XX nahrazuje id konkrétní kategorie. Slouží k vylistování produktů ve vybrané kategorii, které má nastavenou viditelnost.

Odesílané parametry:

- **Id** – id zboží v systému
- **Name** – název produktu
- **Jednotka** – množstevní jednotka
- **Množství** – množství produktu
- **Cena** – cena produktu

/objednavka/creat

URI svázané s REST metodu typu: **POST**.

Určení: Po zavolání vytvoří nový účet k vybranému stolu.

Přijímané parametry:

- **Id stolu** – id stolu, pod kterým je veden v systému

Odesílané parametry:

- **Id** – id objednávky v systému
- **Vytvořeno** – datum a čas vytvoření objednávky
- **Status** – status v jakém se objednávka nachází
- **Stůl** – id stolu v systému
- **Obsluha** – id personálu, který objednávku vytvořil
- **Vytvořil** – nick uživatele, který objednávku vytvořil

/objednavka/list/XX

URI svázané s REST metodu typu: **GET** a **PUT**.

Určení: Zde jsou dvě varianty chování, které se liší podle metody, pomocí které dochází k volání. Ve variantě GET slouží k vylistování objednávek ke stolu. V ukázkové URI XX nahrazuje id stolu. A ve variantě PUT slouží k přejmenování vybrané objednávky a XX v URI nahrazuje id objednávky.

Přijímané parametry ve variantě volání PUT:

- **Popisek** – nové pojmenování objednávky

Odesílané parametry ve variantě volání GET:

- **Id** – id objednávky v systému
- **Vytvořeno** – datum a čas vytvoření objednávky
- **Status** – status v jakém se objednávka nachází
- **Stůl** – id stolu v systému
- **Obsluha** – id personálu, který objednávku vytvořil
- **Vytvořil** – nick uživatele, který objednávku vytvořil

/objednavka/list_polozky/XX

URI svázané s REST metodu typu: **GET**.

Určení: Návratová data jsou tvořena dvourozměrným polem položek, které jsou v objednávce. XX zde v ukázkové URI nahrazuje id objednávky, ke které chceme položky.

Odesílané parametry:

- **Id** – id položky v systému
- **Název** – jméno položky
- **Ks** – počet kusů položky v objednávce
- **Cena** – cena za jeden kus položky
- **Množství** – množství produktu
- **Jednotka** – množstevní jednotka

/objednavka/add_zbozi

URI svázané s REST metodu typu: **POST**.

Určení: Po přijetí dat se do objednávky přidá další produkt. V případě, že se jedná o přidání do fronty, a zboží patří do kategorie jídel, vytvoří se rovnou záznam pro tisk lístku do kuchyně.

Přijímané parametry:

- **Objednávka** – id pod kterou je objednávka v systému
- **Zboží** – id produktu, pod kterým je vedeno v systému
- **Cena** – cena produktu
- **Status** – záznam zda se přidává do fronty, anebo rovnou do objednávky

/objednavka/presun

URI svázané s REST metodu typu: **POST**.

Určení: Zde dochází přesunu položek mezi frontou a dodanými položkami. V případě, že položky stejného typu již v objednávce jsou dodány, dojde k navýšení tohoto počtu a smazání původní položky. Pokud ještě nejsou, dojde k vytvoření položky se statusem dodaný a původní položka se smaže.

Přijímané parametry:

- **Id** – id položky v systému

/objednavka/odecti

URI svázané s REST metodu typu: **PUT**.

Určení: Využívá se v případě, že je potřeba ponížit počet položek v objednávce. Pokud se počet poníží na nulu je celá položka odstraněna z objednávky.

Přijímané parametry:

- **Id** – id položky v systému

/objednavka/smaz/XX

URI svázané s REST metodu typu: **DELETE**.

Určení: K mazání položek z objednávky. XX v ukázkové URI zastupuje id položky, která má být smazána.

/objednavka/transferZuctu/XX

URI svázané s REST metodu typu: **PUT**.

Určení: Slouží v případě, že je potřeba sloučit dvě objednávky do jedné. XX zde zastupuje id objednávky, jejíž položky mají být přesunuty do jiné objednávky.

Přijímané parametry:

- **Kam** – id objednávky do které mají být přesunuty položky

/objednavka/cena/XX

URI svázané s REST metodu typu: **GET**.

Určení: Po zavolání vrátí celkovou cenu objednávky. XX zastupuje id objednávky, ke které chceme znát cenu.

Odesílané parametry:

- **Ks** – počet kusů v objednávce
- **Cena** – celková hodnota objednávky

/objednavka/zaplaceno/XX

URI svázané s REST metodu typu: **PUT**.

Určení: Slouží při placení objednávky. Přijatá data jsou dvourozměrné pole položek, které se z objednávky platí, jelikož nemusí docházet k celkovému uhrazení objednávky. V případě, že dojde k uhrazení všech položek objednávka se automaticky uzavírá.

Přijímané parametry:

- **Id** – id položky

- **Ks** – počet ks položky, které se budou platit

/objednavka/detail_obj/XX

URI svázané s REST metodu typu: **GET**.

Určení: Po zavolání vrátí detailní informace ke zvolené objednávce. XX zastupuje id objednávky, u které chceme detailní informace.

Odesílané parametry:

- **Id** – id objednávky v systému
- **Vytvořeno** – datum a čas vytvoření objednávky
- **Zaplaceno** – datum a čas uzavření objednávky
- **Status** – status v jakém se objednávka nachází
- **Stůl** – id stolu v systému
- **Obsluha** – id personálu, který objednávku vytvořil
- **Obsluha** platba – id personálu, který objednávku uzavřel
- **Vytvořil** – nick personálu, který objednávku vytvořil
- **Uzavřel** – nick personálu, který objednávku uzavřel

/listky

URI svázané s REST metodu typu: **GET**.

Určení: Po zavolání vrátí dvourozměrné pole s daty pro tisk lístků do kuchyně.

Odesílané parametry:

- **Stůl** – id a popis stolu ke kterému je lístek vytvořen
- **Čas** – čas objednání položek
- **Položky** – názvy jednotlivých položek

/nastaveni

URI svázané s REST metodu typu: **GET**.

Určení: Při tisku účtenek jsou potřeba kontaktní údaje daného podniku.

Odesílané parametry:

- **Název** – název podniku
- **Ulice** – ulice kde podnik sídlí

- **Město** – město kde podnik sídlí
- **PSČ** – poštovní směrovací číslo města kde podnik sídlí
- **IČ** – identifikační číslo provozovatele
- **DIČ** – daňové identifikační číslo provozovatele
- **Tel** – telefonní číslo do podniku

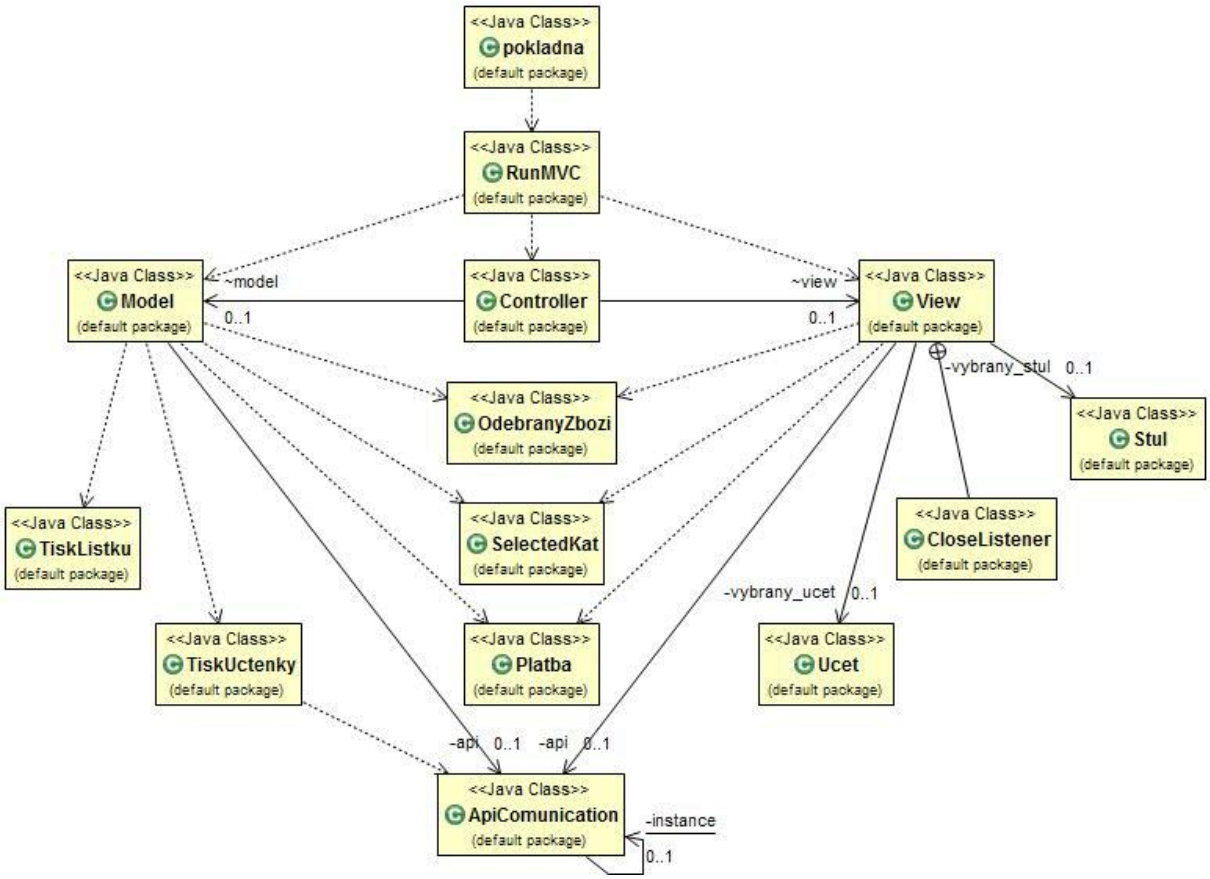
5.3. Pokladna

Aplikace vytvořena v objektově orientovaném programovacím jazyce JAVA s použitím návrhových vzorů a architektury MVC (Model View Controller). K vytvoření této architektury byl zvolen návrhový vzor Observer, který je v jazyku JAVA přímo implementován. Pro komunikaci s API je použita open source knihovna Jersey [3], která umožňuje snadno vytvářet jednotlivé požadavky typu GET, POST, PUT, DELETE. K vytvoření grafického uživatelského rozhraní byla zvolena knihovna Swing, která umožňuje vytvářet klasické grafické elementy (okna, tlačítka, dialogy, atd.)

5.3.1. Soubory a třídy aplikace

Soubory obsahující třídy aplikace:

- *ApiComunication.java* – třída sloužící pro komunikaci s API
- *Controller.java* – základní třída architektury
- *Model.java* – základní třída architektury
- *OdebranyZbozi.java* – třída definující zboží, které se odebírá z objednávky
- *Platba.java* – třída obsahující informace pro výběr položek k zaplacení
- *pokladna.java* – hlavní třída aplikace obsahující funkci main
- *PridanyZbozi.java* - třída definující zboží, které se přidává do objednávky
- *RunMVC.java* – třída registrující objekty dle vzoru observer
- *SelectedKat.java* – třída definující vybranou kategorii
- *Stul.java* – třída popisující konkrétní stůl
- *TiskListku.java* – třída pro tisk lístku do kuchyně
- *TiskUctenky.java* – třída pro tisk učtenky
- *Ucet.java* – třída definující konkrétní objednávku (účet)
- *View.java* – základní třída architektury



Obrázek 14: Základní class diagram aplikace pokladna

Základní kostru aplikace, jak již vychází ze zvolené architektury, tvoří soubory a stejnojmenné třídy Model, View, Controller. Třída Model obsahuje datový model, který upravuje na základě volání z Controlleru a následně data předá třídě View, která již data prezentuje uživateli. Další důležitou třídou, bez které by aplikace nemohla komunikovat s API je ApiComunication. Tato třída implementuje návrhový vzor Singleton. Toto řešení bylo zvoleno z důvodu, nutnosti využívání komunikace ve více třídách.

ApiComunication

Tato třída obstarává komunikaci mezi aplikací a API umístěným na serveru. Při vytváření instance se musí použít platné uživatelské jméno a heslo, jelikož API vyžaduje autentizaci. Vytváří se pouze jedna instance, která je uvnitř třídy a v případě potřeby využití v jiné části aplikace se požádá o její poskytnutí. K vytvoření instance dochází při první žádosti o poskytnutí.

```
//funkce starajici se o poskytnuti instance tridy
public static ApiComunication getInstance(){
	if (instance == null) {
		instance = new ApiComunication();
	}
	return instance;
}
```

Funkce poskytující instanci třídy ApiComunication

Model

Třída Model, má kromě správy datového modelu, za úkol v časových intervalech kontrolovat, zda se nemají vytisknout objednávkové lístky do kuchyně. Tyto kontroly zajišťuje časovač, který se inicializuje při vytvoření objektu této třídy. Při každé periodě je zavolána funkce, které prostřednictvím API zjistí, zda jsou nějaké lístky k vytištění. V případě, že ano vytvoří se objekt třídy TiskListku, kterému se předají data konkrétního lístku, a provede se vytištění. Tisková úloha je automaticky zaslána na výchozí tiskárnu.

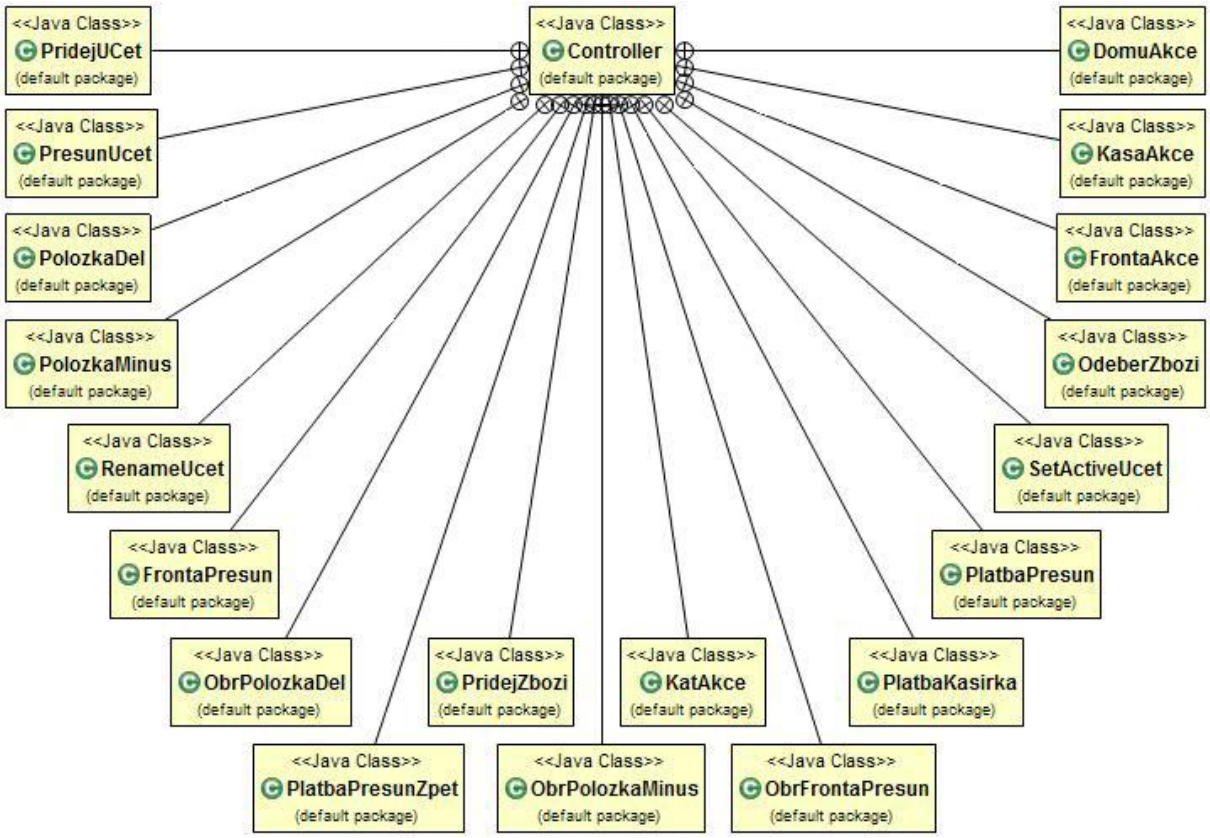
```
	//zpoždění 10 vteřin
	int delay = 10000;
	//perioda opakovaní 30 vteřin
	int period = 30000;

	//vytvoření časovače s nastaveným zpožděním a periodou
	Timer timer = new Timer();
	timer.scheduleAtFixedRate(new TimerTask()
	{
		public void run()
		{
			//volaní funkce pro kontrolu a tisk lístků
			tiskListkuKuchyn();
		}
	}, delay, period);
}
```

Vytvoření časovače pro tisk lístků

Při změně datového modelu z důvodu platby dochází k vytvoření objektu třídy TiskUctenky, který provede vytištění účtenky. Tomuto objektu se předají data o placených položkách. Další nezbytné informace pro tisk účtenky objekt sám získá prostřednictvím třídy ApiComunication.

Controller

Tato třída obstarává řídící logiku. Tedy určuje, co se stane v případě, že uživatel provede nějakou akci, například stiskne tlačítko. Ke každému tlačítku je zaregistrována některá ze tříd, obsažených v controlleru. Každá z těchto tříd implementuje ActionListener (naslouchač událostí) a zachytí odeslané parametry, které předá modelu.



Obrázek 15: Class diagram tříd Controller

View

Zda se vytváří uživatelské rozhraní aplikace. Pomocí knihovny Swing, která je nedílnou součásti jazyku JAVA. Každá obrazovka, je rozdělena do několika panelů. Z důvodu nutnosti aktualizace některých částí na základě uživatelských podnětů je ke každému panelu vytvořena funkce, která se stará o jeho překreslení včetně získání dat. Pro některé panely se data získávají přímo ze serveru prostřednictvím objektu třídy ApiComunication (např. zboží určité kategorie) jindy se zobrazují data, které obsahuje třída Model (např. vybrané položky k zaplacení). Součástí je i sada funkcí, které jsou volány z controleru. Ty k jednotlivým tlačítkům přiřadí jako listener objekt třídy, která je součástí controleru.

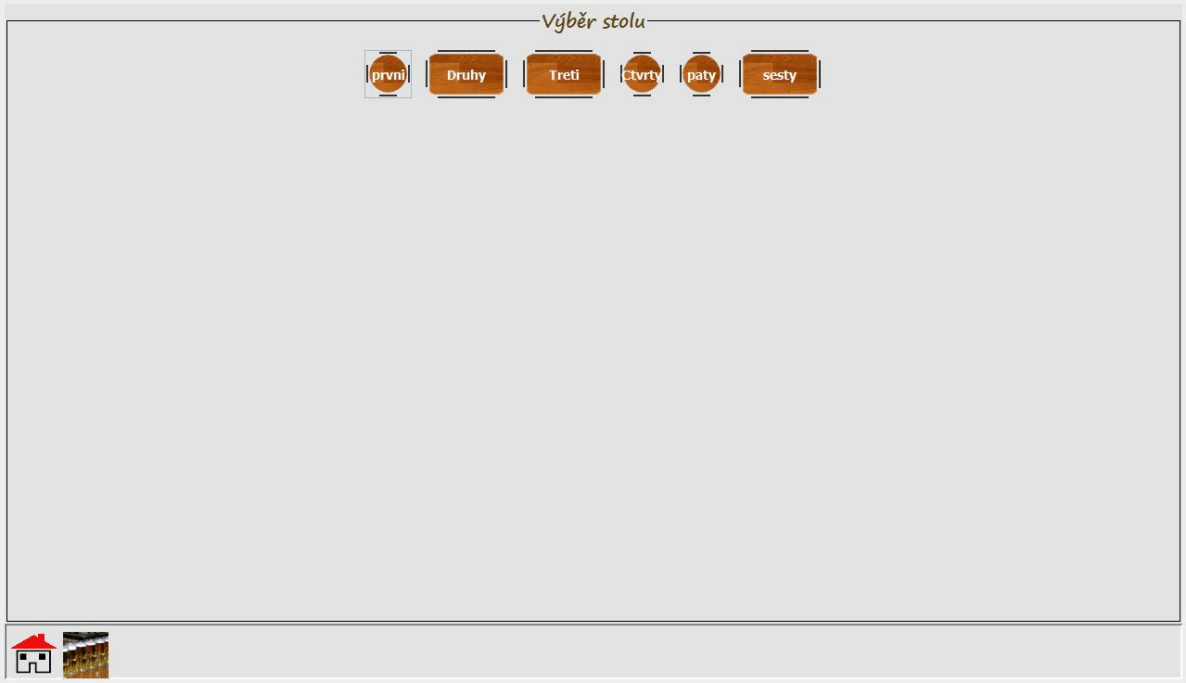
```
//přidání listeneru na jednotlivá tlačítka
public void zaregistruj_tlacitka_pridej_ucet(ActionListener controller){
	//procházení již zaregistrovaných listeneru a jejich odstranění
	for( ActionListener al : pridej_ucet.getActionListeners() )
		pridej_ucet.removeActionListener( al );
	//přidání konkrétního listeneru předaného z controleru
	pridej_ucet.addActionListener(controller);
}
```

Zaregistrování konkrétního listeneru k tlačítku

5.3.2. Obrazovky aplikace

Výběr stolu

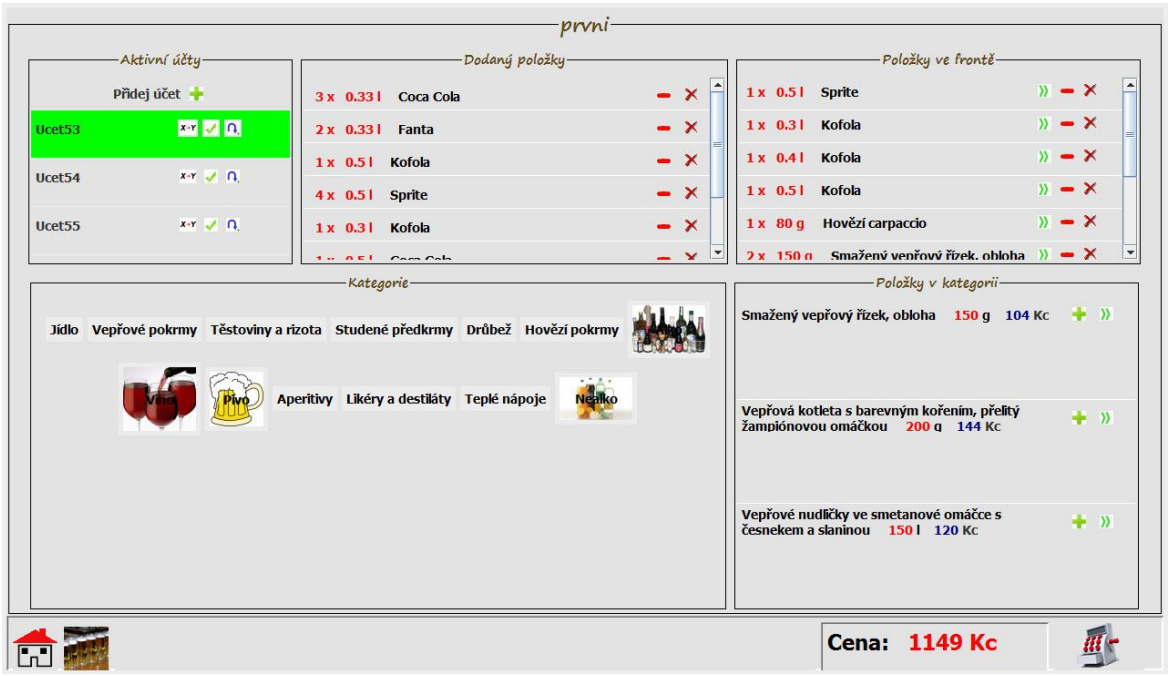
Na obrazovce jsou zobrazeny stoly, které byly definovány ve webové aplikaci. Jednotlivý stůl je reprezentován vybranou ikonou s popisem.



Obrázek 16: Obrazovka výběru stolu

Detail vybraného stolu

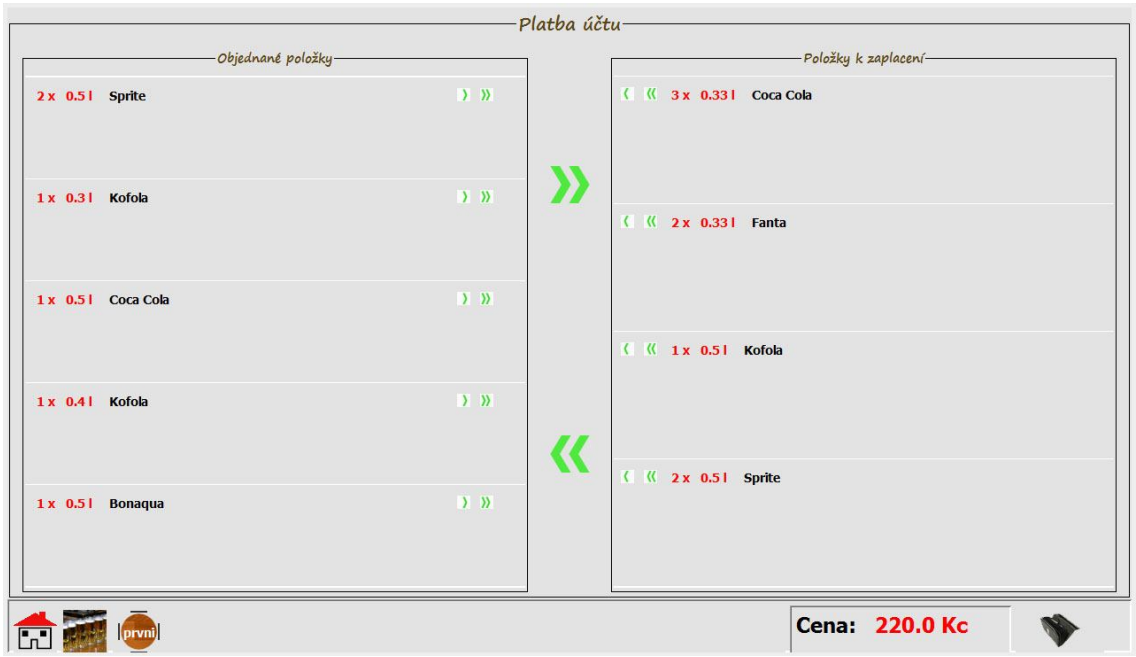
Toto je hlavní pracovní obrazovka, se kterou bude personál pracovat. Je zde možnost spravovat jednotlivé účty zákazníků: vytvářet nové, slučovat, přejmenovávat. Po výběru konkrétního účtu je již možnost přidávat jednotlivé položky, které se zákazník objednal. A buď je zařadit do fronty položek, nebo je rovnou přidat do objednávky. Z fronty, lze po dodání zákazníkovi, položky přesunout do objednávky. Taktéž je zde možnost měnit počet položek, které již byly přidány či je úplně odstranit.



Obrázek 17: Detail vybraného stolu

Platba

Zde již personál vytváří sestavu položek, kterou bude zákazník platit. Lze převést všechny položky najednou, popřípadě pouze určité položky.



Obrázek 18: Obrazovka platby

Fronta položek

Jedná se o speciální obrazovku, která zobrazí všechny položky, které jsou ve frontě bez závislosti na vybraném stolu. Jednotlivé položky lze přesunout do objednávky, popřípadě upravit jejich množství. Také je zde možnost přesunu na detail vybraného stolu.

Obrázek 19: Zobrazení položek ve frontě

5.3.3. Tisknuté lístky

Nedílnou součástí aplikace je i schopnost tisknout na libovolné tiskárně. Tisk probíhá vždy automaticky na výchozí tiskárně definované v systému bez zásahu personálu.

Účtenka

Tento lístek je vytištěn vždy po potvrzení platby.

Testovaci restaurace
Studentská 2
461 17 Liberec
IC: 12345678 DIC: CZ12345678
Tel: 123 123 123

Stul: 1 Ucet #: 53
2013-05-08 11:58:38 Obsluha: pokladna

Polozky v	CZK
1x 0.33l Fanta	25,00
3x 0.5l Bonaqua	60,00
1x 0.5l Sprite	40,00
1x 0.4l Kofola	24,00
1x 300g Špagety CARBONARA, a ...	93,00
1x 80g Hovězí carpaccio	110,00
1x 150g Smažený vepřový říze ...	104,00
Celkem	456,00

DPH	Zaklad	Dan	Celkem
21%	376.84	79.16	456,00
TOTAL CZK			456,00

2013-05-08 13:06:08 Obsluha: martin

Prejeme Vam hezky den

Obrázek 20: Účtenka

Lístek do kuchyně

Jedná se o interní lístek, který slouží pouze k předání informace do kuchyně jaké pokrmy a kam se mají připravit.

STUL: 1 - prvni
CAS: 2013-05-08 12:00:12

Špagety CARBONARA, anglická, v ejce, smetana

Smažený vepřový řízek, obloha

Hovězí carpaccio

Obrázek 21: Lístek do kuchyně

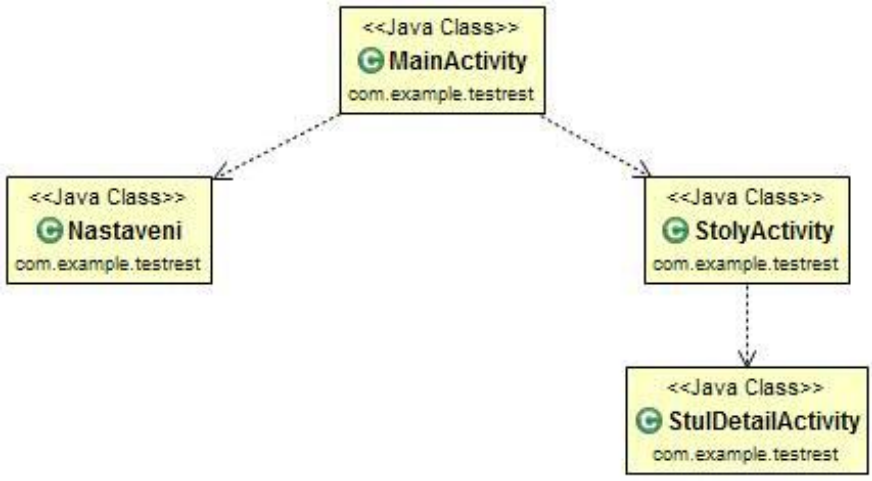
5.4. Mobilní číšník

Aplikace vytvořena pro platformu android. K programování pro tuto platformu se požívá jazyk JAVA, který je upravený pro virtuální stroj s názvem Dalvik. Taktéž byly odstraněny některé knihovny, které JAVA obsahuje, a byly nahrazeny novými speciálně napsané pro android.

Aplikace se skládá celkem ze čtyř aktivit, mezi kterými se uživatel pohybuje a které obsahují třídy využívané v těchto aktivitách. A dvou samostatných tříd, které slouží ke komunikaci. Jedna komunikuje s vytvořeným API a druhá slouží pro komunikaci s databází.

5.4.1. Aktivity:

- **MainActivity** – hlavní aktivita aplikace
- **Nastaveni** – aktivita k uložení uživatelského jména a hesla
- **StolyActivity** – aktivita zobrazující dostupné stolu v podniku
- **StulDetailActivity** – aktivita umožňující obsluhu hosta



Obrázek 22: Schéma aktivit

MainActivity

Hlavní aktivita aplikace, která se zobrazí pří spuštění aplikace. Jedná se o rozcestník k dalším aktivitám (nastavení a přehled stolů). K přechodu dochází při stisku tlačítka.

Nastavení

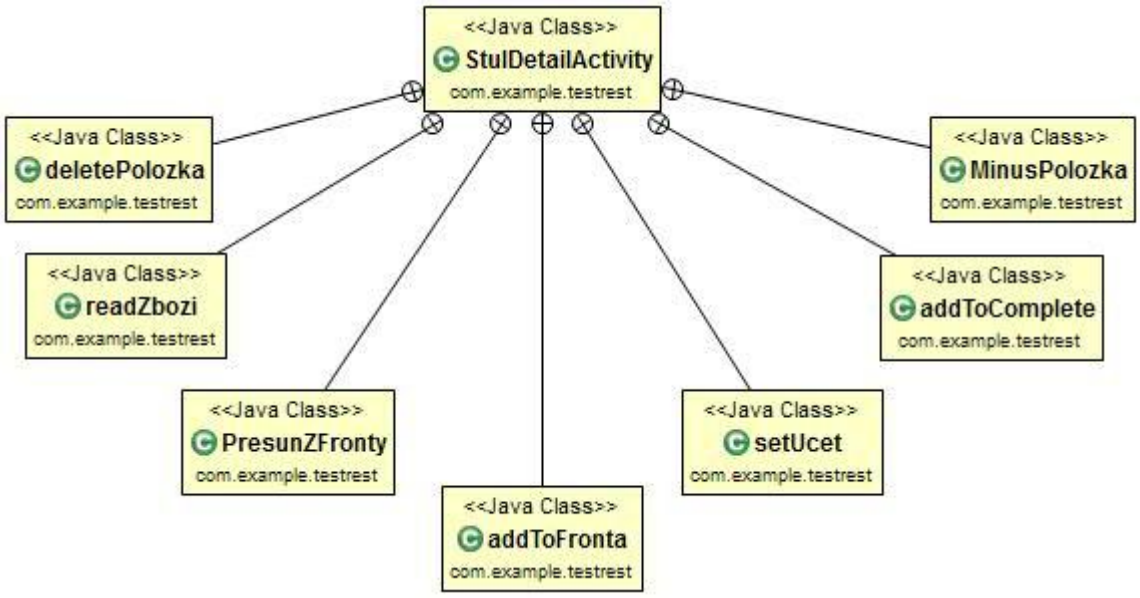
Tato aktivita obsahuje jednoduchý layout, který se skládá ze dvou textových polí (uživatelské jméno a heslo) a dvou tlačítek. Jedno pro uložení jména a hesla a druhé pro resetování hodnot.

StolyActivity

Aktivita zobrazující seznam stolů, které jsou dostupné v podniku. Každý stůl je reprezentován tlačítkem ve tvaru zvoleném ve webové aplikaci a popisem. Při stisku vybraného tlačítka dojde k přechodu na aktivitu, která reprezentuje vybraný stůl.

StulDetailActivity

V této aktivitě již personál pracuje s jednotlivými objednávkami vytvořenými k vybranému stolu. Přidává zde jednotlivé položky do objednávky, upravuje jejich množství, popřípadě je úplně maže. Skládá se z více tříd, které implementují naslouchače k jednotlivým tlačítkům a dvou layoutů. Ty byly nutné z důvodu omezeného prostoru na displeji mobilního zařízení. První layout slouží k zobrazení objednávek u stolu, seznamu kategorií zboží, zboží vybrané kategorie a v úzkém zobrazení dodané položky a frontu objednávky. Druhý layout je rozdělen na dvě části a zobrazuje v detailním provedení frontu a dodané položky objednávky.



Obrázek 23: Class diagram aktivity StulDetailActivity

5.4.2. Samostatné třídy:

- **Db** – třída pro práci s databází
- **RestClient** – třída pro komunikaci s API

Db

Zařízení s platformou Android obsahují databázový systém SQLite. Aplikace tuto

databázi využívá právě prostřednictvím třídy *Db.java*. Tato třída obsahuje funkce pro vytvoření a práci s tabulkou, která slouží pro uložení uživatelského jména a hesla. Pomocí těchto údajů lze následně komunikovat s vytvořeným API.

RestClient

Třída starající se o komunikaci aplikace s API. Jsou zde implementovány funkce pro všechny čtyři typy REST metod (POST, PUT, GET, DELETE). Těmto metodám se předává URI a popřípadě i data, které se mají odeslat a návratové hodnoty jsou ve formátu JSON.

```
public static JSONArray get(String url){
   //vytvoření klientů ke komunikaci
   HttpClient httpClient = new DefaultHttpClient();
   HttpGet httpGet = new HttpGet(url);

   //nastavení http autentizace
   httpGet.setHeader("Authorization", "Basic
      "+Base64.encodeToString((nick+":"+heslo).getBytes(), Base64.NO_WRAP));
   //inicializace proměnných
   HttpResponse response = null;
   JSONArray json = null;
   try {
       //vykonání požadavku
       response = httpClient.execute(httpGet);
       HttpEntity entityResp = response.getEntity();

      //pokud odpověď není prázdná
      if (entityResp != null) {
         //získání a transformování odpovedi do formátu JSON
         InputStream instream = entityResp.getContent();
         String result = convertStreamToString(instream);
         json = new JSONArray(result);
         instream.close();
      }

   } catch (ClientProtocolException e) { Log.d("moje:","chyba1");
        e.printStackTrace();
   } catch (IOException e) {Log.d("moje:","chyba2");
        e.printStackTrace();
   } catch (JSONException e) {
       // TODO Auto-generated catch block
       e.printStackTrace();
  }
  return json;
}
```

Funkce implementující REST metodu GET

6. Závěr

Na základě požadavků společnosti New Prague Media s.r.o. jsem provedl rešerši existujících pokladních systémů. Ze získaných informací o nabízených systémech a jejich vlastnostech jsem vytvořil vlastní návrh systému, který jsem následně implementoval do funkčního prototypu. Ten bude dále sloužit pro vývoj a testování nově vznikajícího systému. Vytvořený prototyp se skládá celkem ze čtyř aplikací:

- webové aplikace - určena pro správu podniku
- API – rozhraní k vzájemné komunikaci aplikací
- pokladna – aplikace pro obsluhu zákazníka
- mobilní číšník – mobilní aplikace k obsluze zákazníka

Každá aplikace obsahuje kompletní funkcionalitu, kterou společnost vyžadovala v rámci této práce. Struktura systému a realizace jednotlivých aplikací je podrobně popsána v páté kapitole.

Budoucí vývoj by se měl zaměřit převážně na rozšíření funkcionality webové aplikace, která slouží provozovateli podniku. Důležitou součástí aplikace by mělo být skladové hospodářství, které umožní další zefektivnění podniku. Rozšířením zobrazovaných statistik lze majiteli usnadnit rozhodování o budoucím směrování jeho podniku. Pro komfort a rychlost obsluhy by bylo vhodné rozšířit funkcionalitu mobilního číšníka na úroveň srovnatelnou s aplikací pokladna.

7. Zdroje informací

[1] FIELDING, Roy Thomas. Architectural Styles and the Design of Network-based Software Architectures. 2000. Doctoral dissertation. University of California, Irvine.

[2] KRAVAL, Ilja. Design Patterns v OOP. 2002. Dostupné z: http://www.objects.cz/

[3] Jersey - API Client. [online]. Dostupné z: https://jersey.java.net/

[4] Dokumentace PHP. [online]. Dostupné z: http://php.net/docs.php

[5] GUTMANS, Andi. Mistrovství v PHP 5. Vyd. 2. Brno: Computer Press, 2007, 655 s. ISBN 978-80-251-1519-0

[6] PECINOVSKÝ, Rudolf. Java 7: učebnice objektové architektury pro začátečníky. Vyd. 1. Praha: Grada, 2012, 495 s. Knihovna programátora (Grada). ISBN 978-80-247-3665-5.

[7] RICHARDSON, Leonard a Sam RUBY. RESTful web services. 1st ed. Sebastopol: O'Reilly, 2007, xxiv, 419 s. ISBN 978-0-596-52926-0.

[8] Android Tutorials. [online]. Dostupné z: http://www.vogella.com/android.html

[9] Android develop documentation. [online]. Dostupné z: http://developer.android.com/develop/index.html

Příloha – A obsah přiloženého CD

Přiložené CD obsahuje adresáře s daty:

- Diplomova_prace – dokument diplomové práce ve formátu PDF
- Mobilni_cisknik – zdrojové kódy aplikace mobilní číšník
- Pokladna – zdrojové kódy aplikace pokladna
- Webova_aplikace – zdrojové kódy webové aplikace